Architettura HSA (Heterogeneous System Architecture) per l'elaborazione delle immagini

L'adozione di un approccio basato su un mix tra CPU e GPU consente di ottenere sensibili miglioramenti in termini sia di prestazioni sia del rapporto prestazioni/W in applicazioni di elaborazione delle immagini

Jonathan Gallmeier
SMTS | CAS
AMD

L'elaborazione dell'immagine è un'operazione che richiede una quantità enorme di risorse in termini di throughput della memoria e della CPU. Il parallelismo, implementato mediante l'uso di più core di CPU, può rappresentare un valido ausilio ma, anche utilizzando i moderni dispositivi a due, quattro o più core, l'esecuzione dei task multimediali richiede la disponibilità di un'elevata potenza di elaborazione o, in alternativa, l'aggiunta di dispositivi hardware dedicati per l'elaborazione delle immagini che possono risultare costosi e comportare quindi una diminuzione dei profitti. L'aggiunta di un insieme di elementi di elaborazione parallela come un acceleratore programmabile alla CPU consente di ottenere un ottimo compromesso tra potenza di elaborazione di tipo general-purpose, elevate prestazioni e bassi consumi. Nel corso dell'articolo verrà dimostrato come l'adozione di una soluzione di questo tipo consenta di ottenere un aumento di un fattore compreso tra due e tre in termini sia di prestazioni sia del rapporto prestazioni/W.

L'elaborazione multimedia ha tratto notevoli vantaggi dalla crescente disponibilità di processori multi-core ma, fino a non molto tempo fa, la tendenza era considerare il concetto di multicore all'interno di architetture di CPU in prevalenza omogenee. Un approccio tipico era quello di rendere disponibili due, quattro o più core organizzati secondo un'architettura SMP (Symmetric Multi-Processor) oppure ccNUMA (cache-coherent Non-Uniform Memory Architecture), talvolta con l'aggiunta di risorse di accelerazione hardware per trasferire l'esecuzione di task particolarmente complessi – come ad esempio la codifica dell'entropia utilizzando gli algoritmi CAVLC o CABAC previsti dallo standard di elaborazione video H.264.

Parecchie società ora sono in grado di offrire funzioni grafiche integrate nei loro dispositivi, principalmente per minimizzare lo spazio a bordo della scheda e i costi della BOM (Bill of Material).

Un altro beneficio legato alla disponibilità di una GPU (Graphics Processing Unit) integrata è dato dal fatto che quest'ultima può essere vista alla stregua di un percorso aggiuntivo per l'esecuzione di task computazionali particolarmente onerosi. Un esempio è riportato in figura 1, che mostra i processori della

Tabella 1 – Parametri di una APU A10 di AMD per applicazioni desktop		
Handbrake A10 5800K (100W TDP)	**CPU**	**GPU**
Freq (GHz)	3.8	0.8
Cores	4	384
FLOPS/core	8	2
GFLOPS	121.6	614.4
A10 5800K Total GFLOPS	736	
A10 5800 Power (W TDP)	100	

serie R di AMD che integrano l'architettura CPU "Bulldozer" con una GPU Radeon discreta (discrete class) per dar vita a un nuova classe di dispositivi che vanno sotto il nome di APU (Accelerated Processing Unit).
Nella figura 1 viene riportato lo schema dell'architettura ad alto livello di questa APU. La GPU è un insieme di elementi di elaborazione di tipi SIMD (Single Instruction Multiple Data). Si noti il controllore di memoria comune tra questo insieme di elementi e i core x86. Questa architettura rappresenta la prima fase di un processo evolutivo verso un'architettura di sistema eterogeneo (HSA - Heterogeneous System Architecture) di AMD che si differenzia in modo significativo dal tradizionale approccio multi-core: esso infatti considera gli elementi di elaborazione grafici integrati del dispositivo come un insieme di risorse computazionali che possono essere utilizzati per dati "normali", ovvero di natura non grafica. Questa modalità di utilizzo è molto simile a quella che prevede l'impiego di un acceleratore hardware connesso: il vantaggio in questo caso è poter disporre di un cluster molto ampio di dispositivo programmabili – fino a 384 core nei dispositivi di fascia alta – che può garantire un throughput di elaborazione di

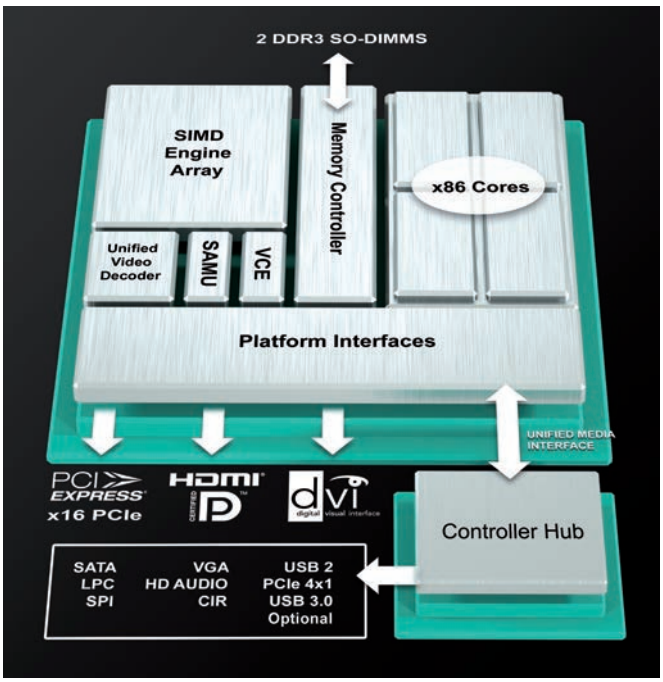

Fig. 1 – Schema di architettura ad alto livello della APU serie R di AMD

Tabella 2 – Dati relativi all'elaborazione dell'utility Handbrake elaborati su un computer desktop equipaggiato con una APU A10

Handbrake A10 5800K (100W TDP)	High Preset, 1080p input transcoded to 720p 6mbps output		
	No GPU	GPU Used + UVD	difference
	14.8 Fps	18.3 Fps	3.5 Fps
	62 sec	48 sec	-14 sec
FloatingPt Ops Total	7539.2	35328	27788.8
Operations/watt	75.392	353.28	277.888

centinaia di GFLOPS (Floating Point Operation per Second). In tabella 1 vengono riportati i parametri di funzionamento base di una APU di classe desktop di AMD (A10-5800K) avente la medesima architettura di base di una APU serie R di AMD. La CPU basata su "Bulldozer"assicura ottime prestazioni in virgola mobile di 121,6 GFLOPS, comunque nettamente inferiori rispetto ai 614 GFLOP della GPU a disposizione per tutte quelle applicazioni in grado di sfruttare in maniera adeguata questa risorsa. La chiave per utilizzare i processori della GPU aggiuntivi in applicazioni di imaging è prendere in considerazione il tempo necessario per trasferire un blocco di dati alla GPU, elaborarli e ritrasmettere i risultati al thread principale.

Sebbene l'attuale generazione di processori serie R di AMD richiede un trasferimento tramite buffer tra GPU e CPU, nelle applicazioni che prevedono l'elaborazione di dati in parallelo possono beneficiare di incrementi in termini di throughput ed efficienza energetica che superano di gran lunga il costo legato alla modalità di trasferimento dati.

Un esempio di applicazione che opera su dati paralleli è rappresentato da Handbrake, una utility molto diffusa per la transcodifica video che può girare su una vasta gamma di sistemi operativi e di piattaforme hardware. Questa utility può vantare un supporto multicore ben integrato e quindi si presta ad essere utilizzata per verificare l'efficacia delle strategie multicore. Handbrake utilizza X264, un codificatore H.264 completo adottato in una moltitudine di prodotti commerciali e open source che godono di un ottimo supporto multi core.

In entrambi i progetti presi in considerazione si è fatto ricorso a OpenCL al fine di utilizzare le risorse di elaborazione della GPU disponibili all'interno della APU. Per quanto riguarda il codificatore X264, è stato effettuato il porting delle ottimizzazione della qualità associate alla funzione di lookahead nell'ambiente OPenCL della GPU. Essa viene utilizzata nelle applicazioni di transcodifica per migliorare la qualità dell'uscita video e di solito "consuma" una percentuale pari al 20% del tempo della CPU. Handbrake usa direttamente X264 come codificatore video. Oltre a ciò, handbrake sfrutta OpenCL per effettuare lo scaling video e la conversione dello spazio colore da RGB a YUV. Per effettuare alcune operazioni di decodifica video si è fatto anche ricorso al decodificatore video hardware (UVD – Universal Video decoder) di AMD. I dati sono stati elaborati su un computer desktop equipaggiato con una APU A10 della serie Trinity che consuma 100W: l'architettura di base è, in ogni caso, identica a quella utilizzata per i prodotti embedded.

L'architettura per la gestione della potenza della APU è particolarmente complessa per cui, al fine di semplificare l'analisi, si è deciso di operare con un TDP (Thermal Design Power) ottimale. I dati evidenziano che quando l'operazione è fatta solamente interamente nella CPU, è possibile ottenere una frame rate (frequenza di quadro) pari a 14,8 fps e il tempo richiesto è pari a 62 secondi. Se nell'elaborazione sono coinvolti i blocchi GPU e UVD, la frame rate aumenta a 18,3 fps mentre il tempo di elaborazione viene ridotto di 14 s (pari al 22%), con conseguente diminuzione dei consumi.

Il totale delle operazioni in virgola mobile riportato in tabella 2 è il prodotto del throughput di tabella 1 e il tempo di elaborazione (121,6 x 62). A questo punto, dividendo per la potenza, è stata creata una figura di merito: operazioni/Watt. Se si esamina il totale delle operazioni in virgola mobile e la figura di merito normalizzata, appare chiaro che le risorse di elaborazione non sono sfruttate al meglio. Il throughput ottenuto utilizzando la sola CPU è di 75 operazioni/W, mentre nel momento in cui viene aggiunta la GPU si ottiene un risultato nettamente migliore: 353 operazioni/Watt. L'incremento del throughput nelle operazioni in virgola mobile è di quasi 5 volte (4,7 volte per la precisione). Una riduzione del 22% del tempo di elaborazione è un risparmio di una certa entità ma, per quanto questa generazione di APU segni un'evoluzione significativa verso un'architettura HSA, tutti gli accessi alla memoria non sono uniformi in termini di latenza e throughput. Quando il kernel accede alla memoria, lo fa in modo diretto. Tali operazioni sono eseguite a una velocità teorica di 22 GB/s - da 16 a 18 GB/s nelle applicazioni pratiche. Quando nella memoria host viene creato un buffer per accedere ai dati della GPU, i dati passano attraverso un percorso differente dove l'ampiezza di banda effettiva è di circa 8 GB/s. Ciò significa che la differenza in termini di ampiezza di banda della memoria tra ciò che il kernel può aspettarsi a livello locale e quello che può aspettarsi se il programmatore imposta un buffer locale nella memoria della GPU è di circa 3:1. Ovviamente si tratta di un processo simmetrico. Se la GPU richiede i dati residenti nella memoria della CPU, la relazione precedente si mantiene inalterata. I futuri dispositivi che implementeranno un'architettura HSA completa saranno caratterizzati da un modello di memoria uniforme, eliminando la necessità di copiare i dati tra le regioni di memoria della CPU e della GPU. Gli algoritmi come le funzioni Lookahead di X264, quando vengono trasferiti

Tabella 3 – Caratteristiche salienti della APU R-464L di AMD		
R-464L (35 W TDP)	**CPU**	**GPU**
Frequency (GHz)	2.3	0.685
Cores	4	384
GFLOPS/core	8	2
R-464L total GFLOPS	599.68	
R-464L power (W TDP)	35	

Tabella 4 - Risultati relativi al benchmark Luxmark 2.0 fatto girare sulla APU R-464L della serie R	
	AMD R-464L (35W)
LuxMark v. 2.0	
Default Integrated benchmark	
OpenCL GPU	198
OpenCL CPU + GPU	230
OpenCL CPU	146

nel dominio di elaborazione della GPU, potranno beneficiare di un'ampiezza di banda di memoria praticamente equivalente.
Un secondo esempio di applicazione che opera su dati in parallelo è Luxmark, un benchmark orientato alla grafica che utilizza OpenCL ed evidenzia i risultati ottenibili quando vengono eliminate le differenze dei tempi di accesso alla memoria vengono eliminate.
Il benchmark integrato con le impostazioni di default è stato fatto girare sulla APU R-464L (della serie Embedded R di AMD) le cui caratteristiche sono riportate in tabella 3.
I risultati riportati in tabella 4 dimostrano in modo netto che la GPU supera in misura superiore al 25% la CPU per questo tipo di compiti di elaborazione. Con un'implementazione HSA completa, ci si potrebbe aspettare che l'aggiunta della CPU alla GPU dia un risultato pari alla somma dei risultati ottenuti da CPU e GPU, ovvero 344. In tabella 4 viene invece riportato un altro numero: 230. Si tratta in ogni caso di un incremento di notevole entità, pari al 63%, ma che non riflette le potenzialità ottenibili con una completa utilizzazione delle risorse. Questa differenza è in gran parte ascrivibile al sovraccarico legato al parsing (l'analisi sintattica) da parte della CPU e al trasferimento continuo dei dati tra CPU e GPU a velocità più basse (come spiegato in precedenza). Minimizzando l'overhead della CPU e della GPU sarà possibile ottenere dall'abbinamento tra CPU e GPU prestazioni nettamente superiori rispetto a quelle conseguibili da ciascuna di esse presa singolarmente.
Le APU di più recente introduzione mettono a disposizione notevoli risorse computazionali a progettisti e sviluppatori di prodotti di elaborazione embedded. Lo sfruttamento di queste risorse richiede la comprensione dell'architettura hardware di base, un esame attento delle problematiche legate al flusso di dati all'interno dell'applicazione target e la familiarità con i tool di più recente generazione come OpenCL.

Bibliografia

A novel macroblock-tree algorithm for high-performance optimization of dependent video coding in H.264/AVC - Jason Garrett-Glaser, Department of Computer Science, Harvey Mudd College