ABSTRACT

The limits imposed by power, memory, and instruction-level parallelism walls have driven the computing industry toward heterogeneous architectures, which combine general-purpose cores with specialized accelerators such as GPUs. These architectures promise higher performance and energy efficiency, but they also introduce significant programming challenges, since efficiently exploiting diverse hardware requires developers to navigate complex vendor-specific APIs and parallel programming paradigms. At the same time, emerging workloads, ranging from AI inference to large scale scientific simulations, demand approximate and energy-efficient computing techniques to meet both performance and sustainability goals. In response, a rich ecosystem of programming models has emerged, spanning low-level, hardware-specific APIs that expose fine-grained control, to high-level abstractions that aim to improve developer productivity while maintaining efficient hardware utilization across heterogeneous systems. While prior research has evaluated low-level programming models, there is a lack of detailed analysis of high-level programming models and their ability to achieve competitive performance compared to low-level APIs, as well as a lack of domain-specific abstractions that expose approximate and energy-efficient computing techniques to developers. This thesis addresses these gaps across the programming-model landscape through three main contributions: the analysis of high- and low-level abstractions for SIMT architectures, the design of domain-specific abstractions and novel techniques for approximate and energy-efficient computing, and the exploration of compiler approaches for automatically generating vectorized code on SIMD architectures. First, we provide a comprehensive evaluation of programming models, assessing how high-level constructs map efficiently to GPUs and approach the performance of native low-level APIs. Second, we extend high-level programming models with domain-specific abstractions and techniques for approximate and energy-efficient computing, enabling developers to

exploit these approaches without specialized hardware knowledge. Lastly, we investigate the autovectorization capabilities of modern compilers for RISC-V vector architectures, providing insights into compiler effectiveness, vectorization coverage, and performance optimization opportunities.

Overall, this thesis contributes to improve the performance of high-level programming abstractions for SIMD and SIMT architectures, as well as the design of new programming-model abstractions and techniques for approximate and energy-efficient computing on heterogeneous architectures.

ABSTRACT

I limiti imposti dalle barriere di potenza, memoria e parallelismo a livello di istruzioni hanno spinto l'industria del calcolo verso architetture eterogenee, che combinano core general-purpose con acceleratori specializzati come le GPU. Queste architetture promettono prestazioni ed efficienza energetica superiori, ma introducono anche significative sfide di programmazione, poiché sfruttare in modo efficiente hardware eterogeneo richiede agli sviluppatori di districarsi tra API complesse e specifiche del fornitore e paradigmi di programmazione parallela. Allo stesso tempo, carichi di lavoro emergenti, che spaziano dall'inferenza AI alle simulazioni scientifiche su larga scala, richiedono tecniche di calcolo approssimato ed energeticamente efficienti per soddisfare sia gli obiettivi di prestazione sia quelli di sostenibilità. In risposta, è emerso un ricco ecosistema di modelli di programmazione, che va da API a basso livello, specifiche dell'hardware e che offrono un controllo fine, fino ad astrazioni ad alto livello volte a migliorare la produttività degli sviluppatori mantenendo al contempo un utilizzo efficiente dell'hardware nei sistemi eterogenei. Sebbene ricerche precedenti abbiano valutato modelli di programmazione a basso livello, manca un'analisi dettagliata dei modelli di programmazione ad alto livello e della loro capacità di raggiungere prestazioni competitive rispetto alle API a basso livello, nonché mancano astrazioni specifiche di dominio che rendano accessibili agli sviluppatori tecniche di calcolo approssimato ed energeticamente efficiente. Questa tesi affronta tali lacune nel panorama dei modelli di programmazione attraverso tre contributi principali: l'analisi di astrazioni ad alto e basso livello per architetture SIMT; la progettazione di astrazioni specifiche di dominio e di nuove tecniche per il calcolo approssimato ed energeticamente efficiente; e l'esplorazione di approcci di compilazione per la generazione automatica di codice vettorizzato su architetture SIMD. In primo luogo, forniamo una valutazione completa dei modelli di programmazione, analizzando come i costrutti ad alto livello possano essere mappati

in modo efficiente sulle GPU e avvicinarsi alle prestazioni delle API native a basso livello. In secondo luogo, estendiamo i modelli di programmazione ad alto livello con astrazioni e tecniche specifiche di dominio per il calcolo approssimato ed energeticamente efficiente, consentendo agli sviluppatori di sfruttare tali approcci senza necessitare di conoscenze hardware specialistiche. Infine, indaghiamo le capacità di autovettorizzazione dei compilatori moderni per architetture vettoriali RISC-V, fornendo approfondimenti sull'efficacia dei compilatori, sulla copertura della vettorizzazione e sulle opportunità di ottimizzazione delle prestazioni. Nel complesso, questa tesi contribuisce a migliorare le prestazioni delle astrazioni di programmazione ad alto livello per architetture SIMD e SIMT, nonché alla progettazione di nuove astrazioni e tecniche di modello di programmazione per il calcolo approssimato ed energeticamente efficiente su architetture eterogenee.