DLP Acceleration on General Purpose Cores

Milovan Duric, Oscar Palomar, Osman Unsal, Adrian Cristal, Mateo Valero
Barcelona Supercomputing Center
{first}.{last}@bsc.es

Abstract- *High-performance and power-efficient multimedia computing drives the design of modern and increasingly utilized mobile devices. State-of-the-art low power processors already utilize chip multiprocessors (CMP) that add dedicated DLP accelerators for emerging multimedia applications and 3D games. Such heterogeneous processors deliver desired performance and efficiency at the cost of extra hardware specialized accelerators. In this paper, we propose dynamically-tuned vector execution (DVX) by morphing one or more available cores in a CMP into a DLP accelerator. DVX improves performance and power efficiency of the CMP, without additional costs for dedicated accelerators.*

I. Introduction

The performance targets of mobile processors are largely driven by the requirements of real time multimedia applications and physical simulations from 3D games. The abundant data level parallelism (DLP) in these applications offers potential for power-efficient acceleration. In this paper we propose dynamically-tuned vector execution (DVX) that exploits DLP on lightweight mobile processors. DVX performs operations over configurable large vector operands, by enabling the resources of general purpose cores to execute vector instructions like classic vector processors. As opposed to classic vector processors that use a fixed number of vector lanes, DVX tunes the size of its execution substrate to a particular workload phase. For this feature, DVX adds lightweight threads of vector instructions controlled by hardware. The threads scale the execution of vector instructions over multiple cores with minimal performance overheads.

II. DVX on General Purpose Cores

DVX morphs the general purpose cores to execute vector instructions that perform compute, register and memory operations over configurable large vector operands. Atomic Instruction Blocks (AIB) with the vector compute instructions allocate the compute resources of one or more cores to perform computation over the vector operands. One instance of such vector AIB is fetched and decoded once, but repeatedly executed multiple times to compute the large operands. The vector operands are accessible only outside the AIB, as the block input or output operands. The vector compute instruction inside the AIB process the operands slice-by-slice in a pipeline, as it is shown in Figure I. By processing large operands in slices (sub-vectors), each new execution of the vector AIB requires only the available

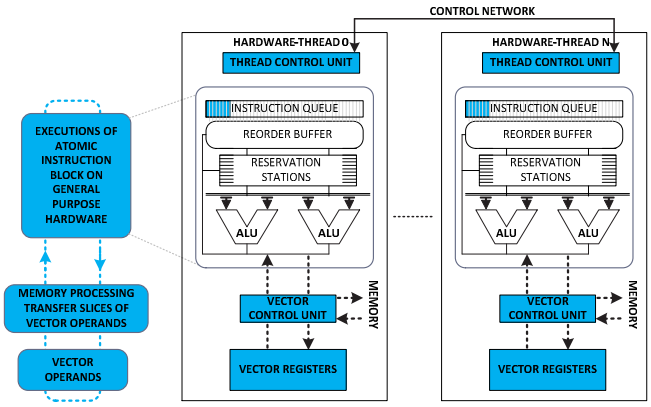

Figure 1. DVX on general purpose CMP utilizes a vector AIB to allocate the existing general purpose hardware of one or more cores that performs the compute operations over vector operands.

general purpose hardware. The hardware processes vector slices as same as scalar operands like in [1] and does not require any modifications. The number of executions is dynamically configured for each vector AIB, to resemble the execution of classic vector instructions that perform one operation over a configurable number of vector elements.

Rather than scheduling vector AIBs to vector cores like in [2], the AIBs leverage the existing issue logic, reservation stations (physical registers) and ALUs of the CMP substrate, as shown in Figure I. The reservation stations keep the temporal results between the instructions in a vector AIB. *Vector registers (VRs)* are an addition to the general purpose core, and they only hold the input/output operands of the AIB. This approach significantly reduces the size of a vector register file, the power-hungry structure that limits the applicability of vector design beyond supercomputer processors. *Vector control unit (VCU)* is a DVX dedicated unit added to the general purpose core to manage DVX. The VCU decouples the execution of vector memory instructions from the executions of vector compute
instructions on the allocated resources. The decoupled memory execution is specialized to tolerate memory latency and provide sophisticated addressing modes required in diverse DLP workloads. Beside memory processing, the VCU reads/writes slices of large vector operands in the VRs and transfers them per execution of compute AIBs.

DVX allows for tuning of its resources to different application requirements by scaling the vector execution over multiple cores. Instead of the existing software approaches, DVX introduces lightweight threads controlled by hardware to dynamically spawn the execution of vector instructions over multiple cores. Such threads avoid startup and bookkeeping overheads in each software thread. DVX adds a simple dedicated *thread control unit (TCU)* to each core of the CMP and an additional on-chip-network between the cores to control the threads with vector instructions.

III. Conclusions

In this paper, we have proposed dynamically-tuned vector execution that accelerates DLP workloads on the set of general purpose cores. The low cost DVX results evaluated in this work makes it a promising alternative to DLP accelerators in commercial processors.

References

[1] Duric, et al. "EVX: Vector Execution on Low Power EDGE Cores." DATE'14

[2] Krashinsky, R., et al. "The vector-thread architecture." ISCA'04

My Publications

[1] "ReCompAc: Reconfigurable Compute Accelerator" RECONFIG'13

[2] "EVX: Vector Execution on Low Power EDGE Cores." DATE'14

[3] "DVX: Dynamic-Vector Execution on a General Purpose EDGE Chip Multiprocessor." SAMOS'14