Work-in-Progress: REDEFINE®™ – A Case for WCET-friendly Hardware Accelerators for Real time Applications

Kavitha Madhu
Indian Institute of Science
kavitha@cadl.iisc.ernet.in

Tarun Singla
Indian Institute of Science
tarun@cadl.iisc.ernet.in

S K Nandy
Indian Institute of Science
nandy@cds.iisc.ac.in

Ranjani Narayan
Morphing Machines Pvt Ltd
ranjani.narayan@morphingmachines.com

Francois NEUMANN
SAFRAN Electronics & Defense
francois.neumann@safrangroup.com

Philippe BAUFRETON
SAFRAN Electronics & Defense
philippe.baufreton@safrangroup.com

ABSTRACT

REDEFINE is a distributed dynamic dataflow architecture, designed for exploiting parallelism at various granularities as an embedded system-on-chip (SoC). This paper dwells on the flexibility of REDEFINE architecture and its execution model in accelerating real-time applications coupled with a WCET analyzer that computes execution time bounds of real time applications.

KEYWORDS

Real-Time Systems, WCET, Multi-core, Parallelism, Dataflow

ACM Reference format:
Kavitha Madhu, Tarun Singla, S K Nandy, Ranjani Narayan, Francois NEUMANN, and Philippe BAUFRETON. 2017. Work-in-Progress: REDEFINE®™ – A Case for WCET-friendly Hardware Accelerators for Real time Applications. In *Proceedings of CASES '17 Companion, Seoul, Republic of Korea, October 15–20, 2017,* 2 pages.
DOI: 10.1145/3125501.3125526

1 INTRODUCTION

Worst Case Execution Time (*WCET*) and Best Case Execution Time (*BCET*) of a real time application are defined in [3]. The terminology defined in [3] lists few other desirable properties namely, *Safeness* that accounts for conservative approximations of uncertainties in the application, and *Tightness* that aims to obtain tighter and precise estimations for the WCET. We present the architecture of REDEFINE with emphasis on its major design objectives namely, performance due to parallelism and predictability. We present a static timing analysis method for good estimates of best and worst case time that meets the *Safeness* and *Tightness* constraints. **REDEFINE Execution Model and Architecture:** REDEFINE [1] is a distributed macro dataflow execution engine for accelerating execution of application kernels (hotspots to be accelerated) specified as some partial order between HyperOps [1]. Each HyperOp is a convex scheduleable data-race free partition of the kernel's data flow graph. Kernels are specified as a HyperOp Interaction Graph (HIG) with vertices corresponding to HyperOps representing computation and edges corresponding to communication. HyperOps are scheduled for execution in strict data flow order. Resources used by a currently executing HyperOp can be used by another HyperOp only on completion of execution of the current HyperOp, eliminating the possibility of context switching within a kernel or across kernels. An abstraction of the hardware model of REDEFINE presenting its major components is shown in figure 1. Context memory holds *context frames* that have placeholders for input data corresponding to the HyperOp they are associated with. Global memory houses the code and operands of the kernel. The three key

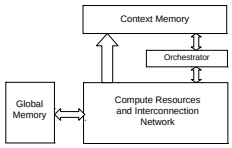

Figure 1: REDEFINE Abstract machine

components of REDEFINE, viz. orchestrator, context memory, and global memory are distributed across the compute resources (CRs) over a single address space on a packet switched toroidal mesh NoC with deterministic XY routing. Each CR houses one or more compute elements (CEs), each of which execute a partition of a HyperOp, referred to as a pHyperOp, created to maximize exploitation of ILP within the HyperOp. Within a CR, CEs communicate among each other via scratchpad memories associated with them, using non-blocking write and blocking read instructions. compiler identifies partitions of HyperOps and inter-pHyperOp communication. A CR also contains a part of the global address space, called the Distributed Shared Memory (DSM). Private L1 data and instruction caches are h each CE and coherence at HIG level is maintained by employing a relaxed dag consistency model. Static analysis is inherently challenging for distributed computing platforms. The design choices of execution model and architecture of REDEFINE ease WCET analysis as follows.

Artifacts of execution model: A fully enumerated HIG for a real time application kernel and its operand data maybe distributed onto a subset of compute and memory resources of REDEFINE statically for optimal performance along with provable execution

CASES '17 Companion, Seoul, Republic of Korea

DOI: 10.1145/3125501.3125526

time bounds. Such HIGs help reason about scheduling order of computations in a distributed setup and also in identifying HIG synchronization points.

Artifacts of architecture: Hardware artifacts of REDEFINE namely memory, network, instruction/data/context frame caches introduce dynamic behavior. Explicit memory management eases estimating memory latencies in a distributed address space.

2 WCET ANALYSIS

Compiler backend offers the flexibility to add instruction latencies as a part of the processor model description. We use the method presented in [2] to create a pipeline model. Modeling cache uses the established *may and must cache analysis* methods. Memory Alias Analysis generated by the compiler is used to analyze a static cache hit. A weighted graph corresponding to interaction between pHyperOps can be constructed statically and the longest path identified to indicate execution time of a HyperOp in isolation. Further, loop analysis needs to be performed to remove cycles induced by loops in HyperOps. Loop scope[2] is a structural representation of loops that represents overlap in loops and identifies the appropriate loop bound to be used in the analysis phase. This information is used to eliminate the back-edges i.e., loop edges in the weighted graph of pHyperOps. Thus, pHyperOps and their parent HyperOp's bounds can be easily computed with known WCET techniques, reducing HyperOp analysis in a larger HIG to that of a HyperOp executing in isolation, owing to the programming model. Local and remote memory accesses come with different latencies. Memory access instructions within a HyperOp are annotated by the compiler to indicate local and remote accesses whenever possible. When the location of data cannot be determined, remote access is assumed when computing WCET whereas BCET assumes an optimistic local access. HyperOps are mapped onto the distributed hardware platform by the compiler such that operands are communicated through the NoC to other HyperOps through context frames. The other form of network traffic originates from remote memory accesses and adds to dynamic memory and operand communication latencies. We compute the BCET and WCET estimates for each HyperOp before NoC analysis. Initially, we employ a naive BCET computation methodology that assume all communication paths to be disjoint and WCET that assumes full overlap in any form of communication. We then employ an iterative algorithm that improves WCET estimates by identifying conflicting BCET-WCET intervals for HyperOps and updates WCET estimates.

3 RESULTS

LLVM based REDEFINE compiler forms the basis for WCET Analyzer tool. The processor model is created using processor itineraries. We evaluate the efficacy of our WCET tool on REDEFINE hardware simulator of size 2×2. Matrix multiplication is performed with a 32×32 block size, with the overall matrix input and output size being 128×128. Fibonacci is a divide and conquer algorithm run for input size 15 without memoization. We observe that the bounds generated by the WCET analyzer satisfies the safeness criterion (figures 2, 3). For a few kernels, tightness is lost due to false paths and poor data locality. Figure 2 presents scalability of the proposed WCET analyzer with increasing number of CEs. Square Root exhibits very limited parallelism keeping its execution time constant. In case of Matrix Multiplication, performing cache analysis per iteration results in identification of a large number of false cache misses, whose effect accrue over iterations. This leads to loss of tightness and can be remedied by identifying steady cache states by unrolling loops during the WCET analysis phase. In FFT(1K), false path contributes to over estimation. Detection and removal of false paths is beyond the scope of this paper. Figure 3 shows how WCET estimation scales further as CRs are increased.

4 CONCLUSIONS

We present a time predictable massively parallel architecture called REDEFINE built with two major design objectives namely, achieving performance by exploiting parallelism at various granularities and WCET friendliness. WCET analyzer is integrated into the compilation flow to make use of the generated *flow facts*. The execution model is data-race free, paving way for parallelism and offering timing composability at a global level. Timing anomalies due to shared network are circumvented through an iterative refinement of the initial bound estimates. The compiler currently use WCET analyzer for proving scheduleability.

REFERENCES

[1] Mythri et. al. Alle. 2009. REDEFINE: Runtime Reconfigurable Polymorphic ASIC. *ACM Trans. Embed. Comput. Syst.* 9, 2, Article 11 (Oct. 2009), 48 pages. DOI: https://doi.org/10.1145/1596543.1596545

[2] Jakob Engblom. 2002. Processor Pipelines and Static Worst-Case Execution Time Analysis. (2002).

[3] Paul Lokuciejewski and Peter Marwedel. 2011. *Worst-case execution time aware compilation techniques for real-time systems.* Springer, Dordrecht, Heidelberg, New York. 3–5 pages. DOI: https://doi.org/10.1007/978-90-481-9929-7

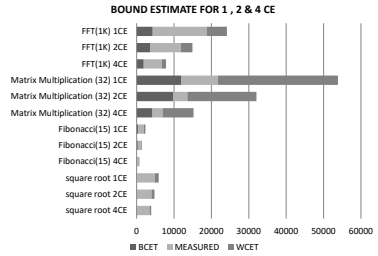

Figure 2: Bound Estimate for 1, 2 and 4 CE

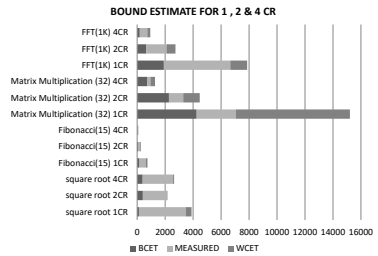

Figure 3: Bound Estimate for 1, 2 and 4 CR