Performance and Memory Access Analysis for Embedded Multi-Core Media Signal Processing Platforms using NoCTrace

Jens Brandenburg and Benno Stabernack

Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute
Image Processing Department, Embedded Systems Group
Einsteinufer 37, 10587 Berlin, Germany
http://www.hhi.fraunhofer.de/

Abstract. Optimizing memory partitioning, memory hierarchy, memory characteristics and allocation of data structures formulates a multidimensional HW/SW co-optimization problem with increasing complexity. This is especially the case for state of the art media signal processing applications running on multi-core platforms with their growing demand for high memory bandwidths. In order to aid the developer with these optimization tasks, performance and memory access analysis tools are needed. Nowadays there exist many different vendor specific debug and profiling tools for different processor architectures addressing different aspects of the overall co-optimization problem. Moving to heterogeneous platforms makes the combination and integration of the different profiling data a challenging task. Moreover it is important to combine the profiling results with information gathered from dedicated components, like interrupts, signals and/or synchronization events, representing the actual hardware platform. To overcome these issues we propose a system level architecture exploration tool called NoCTrace, which uses SystemC as input for the architectural description of the parallel computing system to trace all hardware aspects of the modeled simulation platform in a flexible and generic approach. The tool has been extensively extended to provide comprehensive memory access analysis by gathering cycle accurate bus access statistics combined with detailed program flow information.

Fraunhofer
Heinrich Hertz Institute

Embedded Systems Group

Performance and Memory Access Analysis for Embedded Multi-Core Media Signal Processing Platforms using NoCTrace

Jens Brandenburg and Benno Stabernack, Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute, Germany

Motivation

Goal: Combined run time profiling methodology for parallel computing platforms in respect to HW and SW

Software

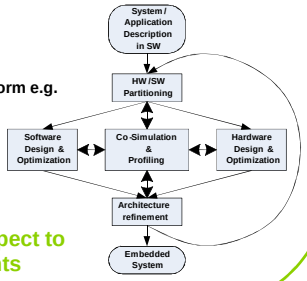
- Find an optimal system level programming model which is suitable for an application specific embedded system
- Find an optimal processor level programming model which is optimized for different processor architectures and applications

Hardware

- Find an optimal system architecture which fulfills the needs of our system level programming model
- Find an processor base architecture which is compiler friendly, extensible and scalable
- Generic approach in respect to actual architecture of the platform e.g.
  - Independent of type and number of processing elements
  - Support all kind of memory system hierarchies / cache / TCM
  - Interconnection networks e.g. Network on Chip

Generic

- Generic in respect to executed software of the platform e.g.
  - Run time system
  - Driver
  - Programming model behavior
  - Memory allocation
  - Run time resource management



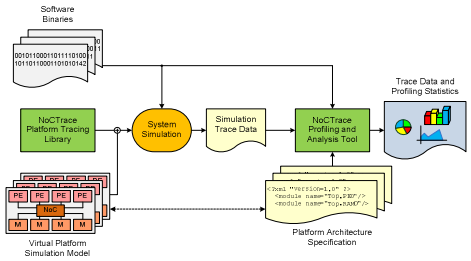
Non-intrusive methodology in respect to software and hardware components

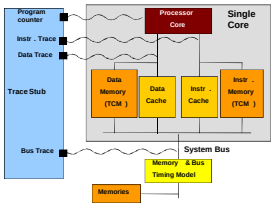
Methodology

SystemC commonly used for the implementation of virtual platform simulation models

Simulation based Tracing and Profiling

- SystemC is a C++ class library providing an event driven simulation kernel (executable is the simulator)
- Implementation of concurrency needed for hardware modeling
- Modeling hardware architectures with different abstraction levels and different timing models / accuracy
- Most used for modeling system-level designs
- Provides simulation kernel
- SystemC kernel + user defined architecture description is executable specification



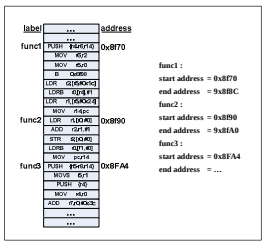


Hardware Tracing

- Bus interfaces to all memory components e.g. TCM, cache
- Timely behavior using clock observation
- System bus interface to external devices
- Program counter of every processing element
- Place probes for any other signals that needs to be observed

Software Profiling

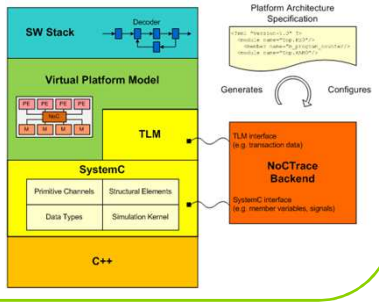
- Use of ISS simulators of used processor cores
- Observation of program counter
- Static analysis of application code
- Generate runtime profile of program counter and associate application code and code map table



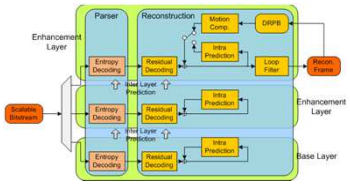
Implementation

Backend and Frontend

- NoCTrace backend simulation task running as a command line task
- NoCTrace frontend integrated into the Eclipse Open Source IDE
- Communication between FE and BE by using TCP/IP sockets
- NoCTrace backend performs an automatic design analysis to generate the platform architecture specification
- XML used as platform specification file format
- Placement of trace data collection probes can be configured by this platform specification
- Trace TLM transaction data, SystemC signals and module member variables

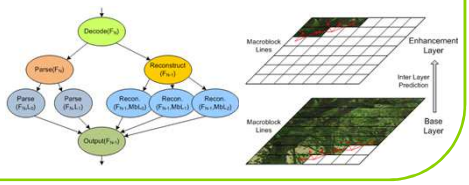


Use Case H.264/SVC Decoder



- Scalable Video Coding (SVC) is an amendment to H.264/AVC
- SVC uses a layered coding approach, where the base layer conforms to H.264/AVC
- Multiple successive enhancement layers can be decoded to increase video resolution, frame rate and/or quality
- Inter layer prediction is used to decrease the combined bit-rate of all video layers

- C code model with optimized signal processing function
- Levels of parallelization:
  - Parallel parser and reconstruction pipeline
  - Parallel parsing of SVC layers
  - Multi-layer wavefront parallel macroblock reconstruction

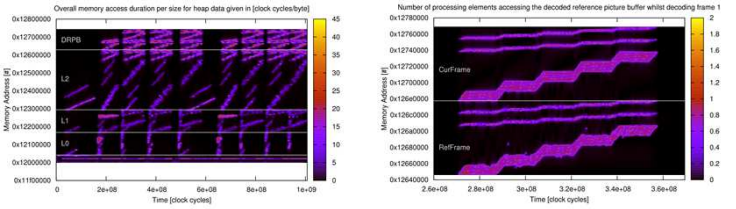


Results

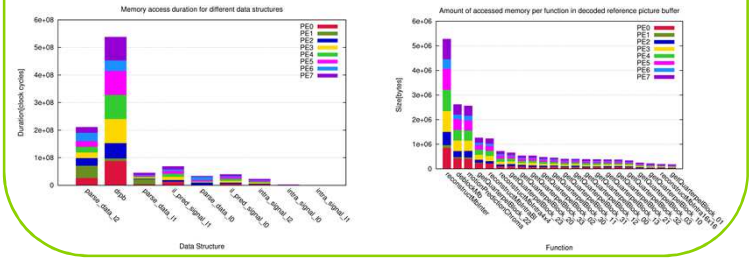
- Platform simulation model (PSM) with 8 MIPS32 ® cores, running 2 parser and 6 reconstruction tasks
- SVC bit-streams contains 3 layers with resolutions: 416x240 (L_0 and L_1), 832x480 (L_2)
- Decode 8 frames using IPPPIPPP
- PSM performs memory accesses during 84.1% of the whole execution time
- Memory access latency varies between: 0.4 cc/byte – 45 cc/byte

Overall PSM execution time	1012.1e^6 cc	
Average memory access duration	851.1e^6 cc	
in instruction memory	284.2e^6 cc	33.4%
in data memory	567.1e^6 cc	66.6%
in function stack	257.0e^6 cc	45.3%
in heap data	264.8e^6 cc	46.7%
in global data	45.4e^6 cc	8.0%
Average memory access size	375.8 MB	
in instruction memory	256.6 MB	68.3%
in data memory	119.2 MB	31.7%
in function stack	71.4 MB	59.9%
in heap data	26.0 MB	21.8%
in global data	21.8 MB	18.3%

Memory Access Pattern Analysis



Fine Grained Function Level Analysis



2PARMA
SEVENTH FRAMEWORK PROGRAMME
FP7-248716-2PARMA Project

www.hhi.fraunhofer.de

Fraunhofer Heinrich Hertz Institute