Abstract: - In this document the Instruction Set Architecture (ASI) simulation issue is discussed. Typical tradeoffs between Flexibility, Speed and Accuracy are shown. Based on hypothesis on Architecture approach (VLIW) and Applications of interest (DSP and Multimedia), this article presents a solution representing a challenging compromise in ASI simulation. A fast, accurate and flexible ASI simulation environment has been implemented using a Simulation cache, innovative pipeline status modeling (Three dimensional), and Simulation Oriented Hw description. Results on two architecture case studies have been reported to validate the described approach: TI TMS320C62x and ST200.

Key Words: VLIW, ILP, Hw-Sw Co-design, DSP, Multimedia, Development Tools, Instruction Simulation, Simulation Speed, Simulation Accuracy.

1 Introduction

The increasing of computational power requirements for DSP and Multimedia application and the needs of easy-to-program development environment has driven recent programmable devices toward VLIW (Very Long Instruction Word) [1] architectures and to Hw Sw co-design environment [2]. VLIW architecture allows generating optimized machine code from high-level languages exploiting Instruction Level Parallelism (ILP) [3]. Furthermore, applications requirements and time to market constraints are growing enormously transferring functionalities from Hardware to Software implementation [4] moving developers toward System on Chip programmable devices. System on Chip application driven design [4] seems to be the answer to fulfill DSP (Multimedia, telecommunication) application complex requirements.

2 VLIW Architectures and DSP Applications

In the previous chapter we introduced ILP to describe how VLIW architectures optimize code execution. There are two possible points of view in considering ILP: the ILP available in a region of code and ILP achievable in the given architecture (Hw-ILP) [5]. The Software developer try to write code with ILP as close as possible the Hw-ILP. Elsewhere Architecture designer analyze the typical application code to match the same Hw-ILP in the Hw design process. Hw-ILP solutions are:

- Multiple Functional Unit to execute at the same time
- Multiple copies of functional units accessing different registers file
- Pipeline for Functional units with latency longer than one cycle

Multimedia and in general digital signal processing applications have typically large available ILP [2], specific data access requirements, repetitive numeric calculation, numeric fidelity, high memory bandwidth and real-time processing requirements [6]. A number of general-purpose processors are suitable for DSP task. Nevertheless DSP processors outperform general-purpose processors in cost-performance rate and power consumption [7] [8]. One of the purpose for Architecture-exploration tools is to allows designers to find the best matching between available and achievable ILP through the interactions between code-development and architecture-parameters tuning tools (Application driven Architectures). Examples of this interaction are Instruction Set or Long Instruction Issue modifications [9].
The approach described in chapter 4 has been used to simulated state of the art VLIW architectures [10][11][12].

3 Instruction Simulation Tradeoffs: Speed, Accuracy and Flexibility

Instruction simulators are nowadays widely used in developing Application-driven architecture design. The architecture design process is strongly influenced by the tradeoff between simulation speed and accuracy: Cycle accurate or close to cycle accurate simulation have usually low performance [13][14]. Better simulation performance can be obtained through vertical software optimization for a given target architecture, but in this case the tradeoff to take into account is between speed and flexibility. Interpretative re-configurable Instruction Set Simulator could efficiently model System on Chip (SoC) and run-time interactions between heterogeneous SoC parts (core, co-processors, etc.). VLIW-SIM ISS is intended to be a Sw application design support and architecture exploration tool.

4 VLIW-SIM: An Innovative Simulation Approach

The VLIWSIM ISA Simulation environment is composed by a set of modules implementing pipeline, memory, Register file, I/D Cache, Instructions and System I/O modeling. Implementation requirement were Interpretive simulation approach, Efficient host memory allocation, Instruction Set Flexibility, Step by step pipeline status tracking, Simulation Speed and Accuracy. Further details on VLIW-SIM environment can be found in [9] [15]. In the following chapters (4.1 to 4.4) the main VLIW-SIM modeling approach is described.

4.1 Pipeline Modeling

The pipeline status is represented in the simulation as a three-dimensional space (phase, operation, time) (Figure 1). The Phase axis represent pipeline phase. The Operation axis represents the instruction position in the Long Instruction Word. The Time axis represent the given time stamp.

The Pipeline status element is the operation internal representation. The simulation process is based on two dimensional arrays representing pipeline status on a given time stamp. Status evolutions take place using two pipeline status arrays representing the current and the following status (Figure 2). The first array’s raw contains the last-fetched Long Instruction Word. The Pipeline Status progression is based on the following algorithm: instructions in a given pipeline phase in the current pipeline status are processed and transferred to the next pipeline phase in the following pipeline status. How the instructions are processed depends on the instruction type and on the phase they are. After all the phases have been updated the current pipeline status is turned into the following.

4.2 Simulation Cache

Performance tests and code profiling have been performed on the simulator. The code profiling has shown that a major over-head is the instruction simulation, and in particular decode and dispatch are the two more frequently executed modules. Moreover typical VLIW applications are repetitive on small piece of code. This feature and the VLIW static scheduling allow introducing the concept of Simulation Locality: large part of code are iteratively simulated, therefore some of the simulation
internal data (e.g. decoded and dispatched instructions) could be re-used, saving simulation time. This idea results in the implementation of a Simulation Cache: a fast access memory based on spatial and temporal locality containing the last N fetch packet with already-processed simulation data. The cache management mechanism algorithm detects hit/miss on cached fetch packets and replaces (following a common block replacement algorithm) dismissible packets. Simulation cache and cache block sizes are definitely critical parameters for simulator performance. To improve significantly the simulated instruction per second (sips) rate simulation cache size should allow critical loop code placement. Measures on a wide set of multimedia applications have been performed to verify locality and to best fit cache and block (cache line) size.

VLIW application locality hypothesis and measures on typical multimedia applications (Figure 3) allow expecting significant improvements introducing the simulation-cache in the simulation environment.

- **Decode Architecture**: From the description, a set of C-like-Macros will be automatically created for each Instruction Mask, in order to extract instruction fields. Moreover each macro has to be associated with the proper entry in the Instruction Field Table (IFT). The IFT is a table containing all the possible meaning (operand, operation code, destination, flag, etc.) for an instruction field in a VLIW instruction, independently from position or size in the mask.
- **Branch Architecture**: The description supply pipeline phases where branch condition is evaluated and target address is computed.
- **Pipeline description**: Pipeline Number of phases and functions per phases (Fetch phases, Decode phases, Execute phases)
- **I/D Cache Parameterization**: Cache Size, Line(Block) Size, number of way (1 is direct mapping), Write on Fetch option, Block Replacement Algorithm.
- **VLIW Parameterization**: Long Instruction Word Size, Register File Size and organization (number of Register Banks and Bank’s size), Control Register File size and organization.

4.4 Instruction Set Dynamic Generation

Instruction Set Flexibility is a major feature to allow Application Driven Architecture exploration and architectural design evaluations [5][16]. The Instruction Set Dynamical generation Tool allows supplying to simulator a behavioral description of the Instruction Set. A language to describe general VLIW Instruction has been identified. For each instruction mnemonic name, instruction class, type of operand(s), destination, latency, operation code and the expression defining the instruction is supplied.

If the instruction uses a control register, the description will contain the field to specify the used register (or parts of the register). Other characteristics can be specified depending on the instruction class. Instructions are divided into three different classes. Each class identifies a specific instruction type: memory operation, arithmetic-logical operation, branches. The description language allows the user to completely define instruction’s behavior through the expression field. In the
expression field, it is possible specify in a C-like notation, all the relations between operands. The user-defined instruction set is taken as input of the Instruction Set Dynamical Generator (ISDG) parser. The parser analyze the description and produces an intermediate instructions representation used to produce the Instruction Set modules.

5 The Simulation Environment
Performance on Two Case Studies

In this chapter test on two VLIW target architectures are reported in order to validate VLIW-SIM simulation environment. The platform used for the test is a Pentium II 400 MHz, 128 MB RAM, Windows NT 4.0 OS. Two different benchmarks have been used: H.263+ [17] coder and G.723.1 [18] [19] encoder and decoder, both implemented in C.

Tests on H.263+ encoder have been performed with the following parameters:
• Test Sequence: foreman.yuv
• Number of encoded frames: 10 (from frame 0 (Intra) to frame 9)
• Video input format: QCIF (176 x 144 pixel)
• Quantization index for P-frame: 10
• Quantization index for I-frame: 10
• Motion estimation search algorithm: Improved Gradient Descent Search
• Motion estimation search window: 15
• Half Pixel Motion Estimation type: subset half pixels

Tests on G.723.1 codec have been performed with the following parameters:
• Test Sequence: ITU standard
• Number of encoded and decoded frames: 20 (from frame 0 to frame 19)
• Audio input format: 8 KHz - 16 bit
• Output Rate: 5.3 Kbps

5.1 Texas Instruments TMS320C62x

In the following test the target architecture is the TI TMS320C62x [10]. VLIWSIM performance and accuracy has been compared with ISS state of the art simulator (HP-ST Lx Instruction Set simulator). The Register files and the memory of the two simulators matched exactly at the end of tests.

5.2 ST-Microelectronics ST200

In the following test the target architecture is the ST Microelectronics ST200 [12]. VLIWSIM performance and accuracy has been compared with ISS state of the art simulator (HP-ST Lx Instruction Set simulator). The Register files and the memory of the two simulators matched exactly at the end of tests.

5.3 VLIW-SIM performance

The following Table resumes the VLIW-SIM performance in terms of Simulated Instruction per Second (sips) in the described case studies for the selected target architectures.

<table>
<thead>
<tr>
<th></th>
<th>VLIW-SIM (ST200)</th>
<th>VLIW-SIM (TI-C62xx)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.723.1</td>
<td>783,752 sips</td>
<td>580,041 sips</td>
</tr>
<tr>
<td>H.263+</td>
<td>724,237 sips</td>
<td>328,731 sips</td>
</tr>
</tbody>
</table>
It should be noted that the ST200 ISS is optimised for a single architecture.

6 Conclusions

In this paper Instruction simulation issues for Hardware-Software Co-Design has been discussed. The Simulation environment VLIW SIM has been presented as solution for VLIW architecture and Multimedia application design support tools. Simulation Locality and Simulation oriented Hardware description were introduced to approach flexibility speed and Accuracy in ISA simulation. Two case studies have been presented to validate the described approach.

Acknowledgments

This research is part of the $^{2}$EDYA project in collaboration with ST-Microelectronics and Hewlett Packard

References


[18] ITU-T Recommendation G.723.1, “Dual rate speech coder for multimedia communication transmitting at 5.3 and 6.3 kbit/s”, October 1995