

Automatic Scheduler for Heterogeneous Multi-pipeline Application Specific Instruction-set Processors

Vathsala Sivalingam, Vijayaratha Vijayasingam

Supervised by: Dr. Swarnalatha Radhakrishnan, Dr. Roshan G. Ragel
Department of Computer Engineering, University of Peradeniya, Sri Lanka

Abstract

In this paper we propose an automatic instruction set scheduler for heterogeneous multi-pipeline application specific instruction set processors. We have implemented this scheduler in Java for ARM-Thumb™ instruction set architecture. According to the application domain to be executed, we can select the suitable number of pipelines in order to increase the performance of the processor. The scheduler will emit instruction and data memory modules which could be used in HDL (Hardware Description Language) simulations.

1. Introduction

Application Specific Instruction-set Processors (ASIPs) are designed for a specific domain of applications. It contains architectural features specific to that domain and still programmable. Therefore they are fast as Application Specific Integrated Circuits (ASICs) and flexible as General Purpose Processors (GPPs). In modern GPPs there are more than one execution units and therefore many instructions can be executed simultaneously. Although this increases the performance of GPPs, they are not optimized for specific tasks due to the generality of the GPPs. In Multi-pipeline ASIPs, each pipeline is customized and implements its own special instruction set, so that the instructions can be executed in parallel with low hardware overhead.

1.1 Background

Research and development has been done in the

area of heterogeneous multi-pipeline processors using ARM-Thumb™ instruction set. A design system has been developed based on the ARM-Thumb™ processor architecture. Given an application specified in C language, the design system can generate a processor model with a number of pipelines specifically suitable for that application. The parallel code for this processor is generated manually and is fed to the relevant pipelines [1].

This design strategy efficiently exploits the instruction level parallelism (ILP) of the application program. In particular, this design contains a novel architecture which tightly couples multiple pipelines via the register file which propose a method to customize the number of pipelines and instruction sets for each of the pipelines; and develop an implementation system with such a design.

1.2 Contribution

In this paper we present an automatic scheduler which we developed using Java for the target architecture specified in [1]. Here we considered all the instructions from the ARM-Thumb™ instruction-set and we scheduled those instructions into a number of pipelines based on the ILP found in the application concerned according to the algorithm given in [1]. Our implementation is user friendly and portable, to which we can input the application program as an assembly file or a C file. Our automatic scheduler will generate the instruction memory and data memory for each pipeline separately. Both these memory modules could be used with our multi-

pipeline processor model for simulation using an HDL simulator such as ModelSim®.

2. Processor Architecture

Our design approach uses the ARM-Thumb™ ISA, which is a cut down version of the ARM™ ISA. The know-how of building a simulation model of a multi-pipeline processor for such an ISA is reported in [1]. Figure1 shows the organization of a multi-pipeline processor.

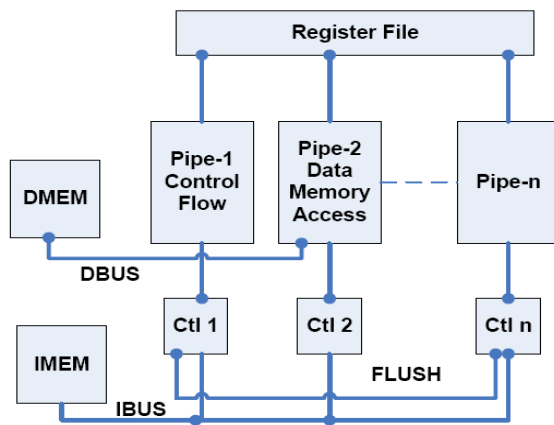


Figure 1: Multi-pipeline Architecture

All pipelines can access the instruction memory. So, they can fetch the instructions parallel at a given clock cycle. The second pipeline (Pipe-2) is connected to the data memory. It transfers the data to register-file and other pipelines, which are able to access the register-file. Branch and compare instructions are assigned to first pipeline (Pipe-1). Instructions which are needed the Data Memory Access Unit (DMAU) are assigned to pipe2. Other instructions can be scheduled into any one of the pipelines.

3. Methodology

Here we describe how we implemented the scheduler. We schedule the sequential assembly instructions, which is the output of cross compiling. As Figure 2 illustrates, we input the sequential assembly or the application program in 'C' to the scheduler and the scheduler will emit the parallel assembly and the relevant binary files for

multi-pipeline simulation. The program is first compiled into a single pipeline assembly code for ARM-Thumb™ ISA.

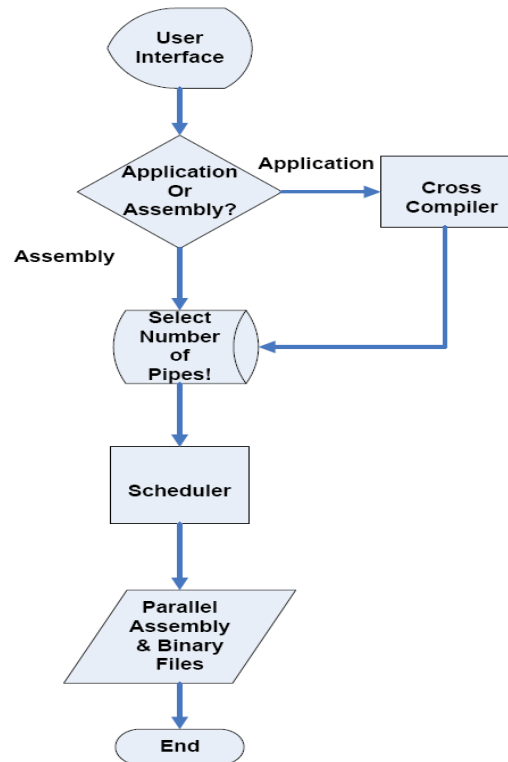


Figure 2: Design Flow

In the next step an initial pipeline number is chosen as the starting search point of the design space exploration. Here user can give his preferred number of pipelines as input and she will be able to get assembly files and the Instruction Memory (IM), Data Memory (DM) files for each pipeline as output. Instructions that are scheduled in the same time slot are executed simultaneously on different pipelines.

3.1 Algorithm Design

The algorithm which we used during the implementation process is as follows. Given the number of pipelines, our scheduler determines the executing pipeline for all instructions such that the performance of the resulting processor is high while the execution time for the block of instructions is minimal. Our scheduling algorithm is based upon the following assumptions and considerations. We need to find a suitable pipeline

for each instruction. For that, we derived some rules to execute an instruction in a suitable pipe.

- Pipe-1 is allocated for control instructions (branch, compare). Pipe-2 is allocated for memory instructions such as load, store, push, etc. Other pipes are capable of executing any other instructions except the two kinds which are mentioned above.
- The lowest level at which each instruction can be put into a pipe is decided according to the dependencies of each instruction within a basic block. Finding dependencies of each instruction involves several restrictions and conditions [2].
- Here we have implemented the scheduler for predefined pipelines which function as homogeneous pipelines. To make them to work as heterogeneous pipelines, we can manually select the functional units for each pipe and can increase the performance while considering the area cost of the hardware for each pipeline.

4. Results

An automatic instruction scheduler for multi-pipeline processor, which is capable of scheduling instructions in effective manner for ARM-Thumb™ ISA is implemented and tested. Our scheduler gives parallel assembly and the relevant binary files for each pipeline. The input application to our scheduler could be a 'C' program or an assembly program and user has to select the number of pipelines.

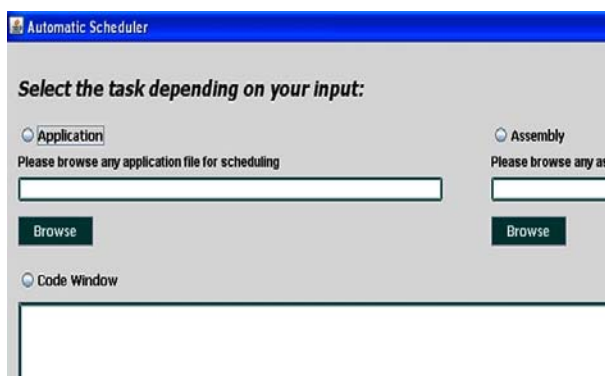


Figure 3: User Interface for Automatic Scheduler

Figure 3 shows the user interface of the Automatic Instruction Scheduler. Here three options are available to user to input her application: (1) loading a 'C' program, (2) loading a sequential assembly program and (3) writing her own application on 'C' in the given text box.

5. Conclusion

Our approach relates instruction level parallelism with the multiple pipeline architecture. An effective parallel instruction scheduling algorithm is used to determine the number of pipelines and the instruction sets to be implemented by each of the pipelines such that the high performance improvement can be achieved [1].

Here we have implemented an automatic instruction set scheduler, which is user friendly and it uses an effective and efficient scheduling algorithm. User can input the application in 'C' or assembly file directly as an input to the scheduler and will be able to get parallel assembly files and the binary files for each pipeline separately for ARM-Thumb™ ISA.

Performance improvement of the processor is achieved by specific instructions (the related work and approach can be found in [1], [3]), improved pipeline structure, and parallel instructions executing on the multiple pipelines.

After getting the ISA of each pipeline we can manually select the hardware components for each pipe.

6. References

- [1] *Customization of Application Specific Heterogeneous Multi-Pipeline Processors*. Swarnalatha Radhakrishnan, Hui Guo, Sri Parameswaran,
- [2] *ARM Architecture Reference Manual*
- [3] S.Radhakrishnan, H. Guo, and S. Parameswaran. *Dual-pipeline Heterogeneous ASIP Design*. IEEE Computer Society, 2004.