UNIVERSITAT POLITÈCNICA DE CATALUNYA

LOOP PIPELINING WITH RESOURCE AND TIMING CONSTRAINTS

Autor: Fermín Sánchez

October, 1995

ACKNOWLEDGMENTS

I would like to thank the members of the Department of Computer Architecture for their support throughout the development of this work. In particular, I would like to thank Jordi Cortadella for his guidance and support throughout my graduate career. His enthusiasm for my early steps in the field gave me the confidence to pursue my own ideas. Besides being my advisor, he has been my best collaborator all these years. I would also like to give special thanks to Rosa M. Badia for her suggestions and comments, which have contributed to the improvement of this work. My gratitude goes also to the rest of the CAD-VLSI group. They have also contributed by their constant encouragement for me to finish this work.

I would like to thank my colleagues in the DAC, especially Anna del Corral, Josep LLosa, Angel Toribio, Mildred Sarmiento, Enric Pastor, Agustín Fernández and Montse Peiron. They have made my years in the University much more pleasant. From among all of them, my deepest gratitude goes to Josep LLosa for the great quantity of discussions that we have maintained in recent years, which have doubtlessly contributed to the enrichment of this work.

I thank Marc Noi for his help with the Farey's series, and Tricia for being my English advisor all these years. I also thank Tomás Lang, David Padua and Mateo Valero for giving me part of their valuable time, listening to my ideas and giving me their suggestions. I am equally grateful to Q. Ning, R. Govindarajan, Eric R. Altman and Guang G. Gao for supplying me the data dependence graphs used for comparisons in superscalar and VLIW processors.

Finally, I am greatly indebted to Ivette, who has always been understanding about my work. I would like especially to thank my brother David and my parents Herminio and Francisca. Their love and support have given me the courage to finish this work. I am privileged to belong to such a wonderful family. This work is dedicated to them.

To my family, the best in the world

CONTENTS

LIST OF FIGURES

Chapter 5

Chapter 6

Chapter 7

LIST OF TABLES

Appendix B

Appendix C

LIST OF ALGORITHMS

PREFACE

This work presents three algorithms to solve three different problems:

- *UNRET* is proposed to solve loop pipelining with resource constraints.
- *TCLP* is proposed to solve loop pipelining with timing constraints.
- *RESIS* is proposed to reduce the number of registers required by a schedule.

Loop pipelining with resource constraints can be defined as follows: "given a set of resources, finding a pipelined schedule of a loop in the minimum number of cycles". *Loop pipelining with timing constraints* can be defined as follows: "given a maximum time to execute an iteration of a loop, finding a schedule which requires the minimum set of resources (or the minimum area)". Whilst loop pipelining with timing constraints is a typical problem in the high-level synthesis of VLSI circuits, loop pipelining with resource constraints is present in both the high-level synthesis of VLSI circuits and compilers for parallel architectures.

In parallel architectures, the number of registers available to store partial results (during loop execution) is limited, and it is defined by the architecture. In a VLSI circuit, a register consumes space in the chip. Therefore, it is interesting in both areas to obtain a schedule which requires as few registers as possible.

UNRET and *TCLP* are related to the extraction of the parallelism of a loop. Chapter 1 introduces different ways to exploit such a parallelism, as well as the subjects on which this work is focused: high-level synthesis of VLSI circuits and compilation techniques for superscalar and VLIW processors.

UNRET and *TCLP* belong to a family of techniques known as *software pipelining*. An overview and classification of such techniques is presented in Chapter 2.

In Chapter 3 we define two transformations to exploit the parallelism in a loop: *dependence retiming* and *loop unrolling*. Both transformations will be used by *UNRET* and *TCLP*. The maximum parallelism available for exploitation in a loop is limited. Chapter 3 also shows how this maximum parallelism can be calculated.

A data dependence exists between two instructions when the result produced by the first one is consumed by the second one. Data dependences impose a partial execution order in the instructions of a loop. According to whether a data dependence does not influence in the scheduling, or it influences the scheduling within an iteration or across consecutive iterations, we classify data dependences into three categories: *free scheduling dependences*, *positive scheduling dependences* and *negative scheduling dependences*. Chapter 4 presents the theory behind this classification.

Chapter 5 describes the scheduling algorithm used by *UNRET* and *TCLP*. The algorithm takes resources into account, as well as multiple-cycle (possibly pipelined) functional units and instructions that have complex execution patterns (they use several functional units during several cycles).

Chapter 6 presents *UNRET*. *UNRET* uses *dependence retiming* and *loop unrolling* to find a pipelined schedule of the loop. *Loop unrolling* is in general required to extract the maximum parallelism. *Dependence retiming* enables us to obtain different (but equivalent) configurations for the same (possibly unrolled) loop. Each configuration is scheduled by using the algorithm presented in Chapter 5, attempting to find a schedule which executes the loop with the maximum parallelism. When no schedule exists for any configuration of the loop, *UNRET* decides a new target parallelism (and unrolling degree) and explores new configurations.

Once a schedule has been found, the number of required registers is reduced while maintaining the parallelism. In Chapter 7 we propose *RESIS*, an algorithm oriented to such a purpose. *RESIS* works in two phases. First, several configurations of the loop are explored, attempting to reduce the number of different iterations involved in the pipelined schedule. Each configuration is independently scheduled by using the algorithm from Chapter 5. Following this, some instructions are individually rescheduled in order to reduce the register requirements.

Chapter 8 presents *TCLP*, an algorithm for loop pipelining with timing constraints. *TCLP* is based on ideas similar to *UNRET*. The timing constraint is given in the form of a maximum number of cycles to execute each iteration of the loop. *TCLP* analytically calculates a minimum set of resources (theoretically) required to execute the loop with the given timing constraint. Several configurations of the loop are explored in order to find a schedule by using the calculated set of resources. If no schedule is found, the set of resources is successively increased and new configurations are explored until a schedule is found. Once a schedule fulfilling the given timing constraint has been found, *TCLP* attempts to optimize several characteristics of the schedule. First of all, *TCLP* attempts to reduce the set of resources while maintaining the length of the schedule. Following this, it attempts to increase the parallelism of the schedule by exploring different unrolling degrees. Finally, *RESIS* is used to reduce the number of required registers.

Chapter 9 presents the conclusions of this work, summarizes the main contributions performed and indicates futures areas of work.