CHAPTER 1 INTRODUCTION

Significant advances in VLSI process technology have scaled the feature size down. As the transistor size shrinking, more and more transistors can be integrated into a single chip. To make large number of components in a chip working together well, processor-based design methods have become the best choices. In a processor design, power consumption and performance are two important issues.

In recent years, the major design considerations of microprocessors have focused on power consumption. Among system components, fetching and decoding instructions consume a large percentage of system power [1][2][3]. For example, the study in [4] shows that fetching and decoding instructions consumed approximately 45% of system power in *StrongArm*. As a result, the reduction of power consumption of fetching/decoding instructions is very important.

Power consumption has also become one of the most important design issues for DSP designs targeted to multimedia and handheld applications. An important trend for low power DSP designs is to customize an instruction set for accommodating program characteristics with ASIPs (Application Specific Instruction-set Processors) [5]. In ASIP designs, due to the high power dissipation, multipliers are the most critical components in an application-specific design.

One effect of the process technology scaling down is that coupling capacitances have grown reciprocal in the square of the scaling factor. This crosstalk effect will not only increase the power consumption but also lengthen the propagation delay. According to the report from NTRS [6], the processor speed will achieve *2* (or *3*) Ghz in 0.07 um technology. That means the instruction fetch stage should be no longer than *0.5* (or *0.33*) ns. On the other hand, the author of [7] pointed out that an

optimized delay of system buses will take *0.67* ns in 0.07 um technology based on the report from NTRS. In another word, the fetch stage will be the bottleneck of a processor when taking into account the delay time of instruction bus, memory access, and address bus in the fetch stage. Hence, the reduction of delay time of instruction bus, memory access, and address bus becomes very important.

The mentioned power and performance issues are related to the execution of an instruction. During the execution of an instruction, three main steps are fetching, decoding, and executing. We will propose techniques to improve performance and power consumption in these three steps.

First, for the step of fetching instructions, we proposed post-compiler optimization algorithms for improving performance of fetching instruction. One effect of the process scaling down is that coupling capacitances have grown reciprocal in the square of the scaling factor. This crosstalk effect will not only increase the power consumption but also lengthen the propagation delay. The coupling capacitance between adjacent neighboring wires, such as buses, on the same metal layer induces a very large fraction of the total capacitance. As a result, how to solve the crosstalk problem on buses has become an important issue.

Most of the previous proposed crosstalk-eliminating techniques were all performed at logic level. They have no knowledge on transition sequences and hence assume that all possible sequences will appear on the bus. Hence, the area overhead for codec logic to eliminate crosstalk is very high. We found that the data sequences on an instruction bus are known during the compile time. This motivates us to study how to eliminate crosstalk effect on an instruction bus for performance improvement using the post-compiler optimization algorithms. Since the elimination techniques are performed at post-compiler level, no hardware (area) overhead is needed. In this thesis, we present two algorithms, instruction rescheduling and register renaming, to

eliminate the crosstalk effects on an instruction bus.

Second, for the step of decoding instructions, we present two techniques for decomposing instruction decoders to minimize power consumption. First, based on the irregular execution frequency of instructions, we propose to decompose an instruction decoder into two or more coupling sub-decoders, a process called horizontal decomposition. With this approach, only one small sub-decoder is activated at a time. Second, based on a pipelined decoder structure, we partition the decoder into two pipelined stages, a process called vertical decomposition. The first stage identifies instruction types, while the second stage generates control signals. With this approach, only the second stage is activated in the subsequent execution stages.

Finally, for the step of executing instructions, we proposed techniques to improve the multiplication power consumption. In [8], a configurable structure was proposed to reduce the power consumption of multiplier designs. The key idea of a configurable multiplier-structure [8] to save power consumption is that the multiplier has two configurations. When the smaller multiplication is performed, unused parts of the multiplier are turned off, where "turn off" means gating input signals. This technique has been proven to be very effective in power reduction. Nevertheless, these techniques focus on ASIC (Application Specific Integrate Circuit) designs rather than processor designs. Moreover, the bit-width of the smaller configuration is simply chosen as the half bit-width of the larger multiplier.

However, we have observed that a smaller configuration with half of the maximum bit-width is not necessarily a good choice for the bit-width distribution of multiplication instructions for some specific applications. We will present a technique to exploit the architecture [8] and determine the bit-width of multiplication instructions.

The rest of the thesis is organized as follows. Chapter 2 presents the related work.

Chapter 3 presents the performance-driven crosstalk eliminating algorithms. Chapter 4 presents the decomposition techniques for low power instruction decoder designs. Chapter 5 presents the techniques for the power-driven multiplication instruction-set design method. Finally, Chapter 6 presents the conclusions and future work.