# CHAPTER 6 CONCLUSIONS AND FUTURE WORK

In this thesis, we have presented three techniques for high performance and low power processor designs. Three techniques are developed for the steps of fetching, decoding, and, executing.

For the step of fetching instructions, based on pre-designed and low-crosstalk instruction op-codes, two post-compiler optimization algorithms, rescheduling and renaming, are presented for performance improvement on an instruction bus by eliminating crosstalk. For rescheduling, we eliminate crosstalk by first modeling the instructions in a basic block to a weighted and directed graph and then applying a TSP algorithm to determine a new instruction sequence. For renaming, we eliminate crosstalk by first modeling the relation between register indexes as a weighted and undirected graph and then applying a simulated annealing based algorithm to rename registers. Since the elimination techniques are performed at post-compiler level, no hardware (area) overhead is needed. The results show that our crosstalk-eliminating post-complier algorithms significantly reduce the $4 \cdot C$ crosstalk sequences from 11.50% to 0.52%. Due to the elimination of $4 \cdot C$ crosstalk, our proposed method can improve the instruction fetch time up to 9.59% with small overhead of power and static instruction count.

For the step of decoding instructions, we have presented two techniques, horizontal and vertical decomposition, for decomposing instruction decoders for low-power designs. For horizontal decomposition, we decompose an instruction decoder into two or more coupling sub-decoders based on the non-uniform execution frequency of instructions. With this approach, only a small sub-decoder is activated at

a time. For vertical decomposition, we partition the decoder into two pipelined stages based on a pipelined decoder structure. The first stage identifies instruction types and the second stage generates control signals. With this approach, only the second stage is activated in the subsequent execution stages. We implement both techniques and conduct three sets of experiments. The results show that our proposed techniques provide effective power reduction with small area overhead. In addition, our proposed decomposition techniques also reduce the length of the critical path.

For the step of executing instructions, in this thesis, we have presented a novel power-driven multiplication instruction-set design method for ASIPs. Using a dual-&-configurable-multiplier structure, we have presented a multiplication instruction-set formation algorithm to generate instruction set by first modeling a multiplication instruction execution sequence into transition graph and then applying our algorithm to determine the bit-width of a dual-multiplier and configurations in each multiplier. The experimental results have shown that our proposed instruction-set formation method can generate a power- and execution time-efficient multiplication instruction set.

Based on the above research results, our future work includes the following two directions.

(1) The proposed post-compiler optimization algorithms (Chapter 3) are for a single-instruction-fetch architecture. However, for most high performance systems, such as *Super Scalar* and *VLIW*, a multiple-instruction-fetch architecture is adopted. A future work on extending the post-compiler optimization algorithms to handle multiple-instruction-fetch architecture needs to be studied further.

(2) Based on the observation of uneven execution frequency, the techniques for the decomposition of instruction decoder (Chapter 4) generate low power instruction decoders. As described in Chapter 4, the more uneven the instruction execution

frequency is, the less the power consumption is. To make the distribution of instruction-execution frequency more uneven, we should explore compiler techniques to take the instruction execution frequency into consideration.