CHAPTER 6: CONCLUSION AND FUTURE WORK

This work proposed the new dimension of extending the usage of FPGA as a co-hardware unit by software professionals in par with the general-purpose processors to improve the execution speed of computation intensive applications. At the same time, the active power consumption is reduced at the cost of interfacing the FPGA as a co-hardware unit as it requires only initial configuration to execute without any phases like instruction fetch, decoding etc. In order to have the core benefit of FPGA, an efficient scheduling mechanism is proposed, designed and implemented.

The first step of implementation is the problem of identifying the potential software codes from the application implementation that can be converted to hardware circuits or CCHBs. It is achieved through application profiling and graph matching modules. The second step is the selection of best CCHBs that are independent of each other. It is framed as the problem of determining the maximum independent set from the dependency graph of potential CCHBs and solved using conflict graph approach. In the third step, the optimization is carried out to reduce the FPGA cell occupancy and thereby maximizing the utilization of FPGA. It is resolved by identifying the same operations of more than one CCHB and combining them into a single merged CCHB using compatibility graph approach.

Five different applications from different domains are individually implemented and compared using the proposed method, Verilog hardware definition language with FPGA alone and C code with conventional microprocessor or microcontroller alone. The number of logic cells for proposed method implementation is compared with complete
FPGA implementation and the results have proved by occupying fewer cells. Similarly the execution speed of the proposed method is evaluated with microprocessor implementation and the outcome provided evidence for improved speed. Further the five different applications are combined to form 2 combinations.

The proposed scheduling mechanism applies to the combinations of application systems. The results are tabulated, tested with an optimization tool and found to agree. The analysis is also done on the results to study the behavior of an application when scheduled along with similar and different applications. The proposed scheduling mechanism is also compared with random order and first come first served mechanisms and it has proved its efficiency.

This work can be enhanced in the future by including priorities and deadlines to the attribute list of CCHBs when subjected to scheduling. The CCHBs are identified based on the templates that are pre-stored. Learning cum automatic update of template library can be designed and implemented, as and when the applications are making use of that library.