Chapter 12
Conclusions and Future Directions

In this chapter, we summarize the major contributions of this thesis and show how the NoC design methods and reliability mechanisms are integrated in the design flow.

12.1 Putting It All Together

The reliability enhanced Netchip tool flow is presented in Figure 12.1. Initially, along with the application traffic characteristics, the system reliability specifications and requirements are also taken as inputs to the tool flow. In Chapter 11, we had presented the mechanisms to provide NoC support for using back-up memories. The number of back-up memories used and the additional traffic flow rates related to them are given as part of the system reliability specifications. In the Netchip flow, we automatically design the NoC to meet the bandwidth demands of the additional traffic that is generated due to the use of multiple memories. The tool flow also ensures that the traffic streams to the back-up memories do not create deadlocks with the other traffic flows.

We had presented methods to achieve tolerance against temporary errors in Chapters 8 and 9. The *T-error* scheme presented in Chapter 8, is required when multi-bit timing errors can occur in the system. Based on the error-rates of the system, which is given as part of the system reliability specifications, we determine whether *T-error* scheme is needed for the NoC. We also determine the most power optimal encoding needed to tolerate other transient errors in the system (such as soft-errors). To determine the best scheme, we use the analysis method presented in Chapter 9.

Once the error recovery schemes needed for the NoC components are determined, we proceed with the design of the NoC topology. The systems that utilize NoCs can be broadly classified into two types: *Application-Specific Systems-on-Chip* (ASSoCs) and *Chip Multiprocessors* (CMPs). In ASSoCs, single or a fixed set of applications are statically mapped onto the different processor and hardware cores in the design. In CMPs, software tasks are dynamically assigned to the cores. We distinguish three major application classes for NoCs here: (1) ASSoCs that run a single application, (2) ASSoCs that run multiple applications, and (3) CMPs that run general software tasks.

For designing ASSoCs that run a single application, we apply the SUNMAP and SUNFLOOR tools presented in Chapters 4 and 5. The SUNMAP tool is used to design a standard topology (such as a mesh, torus) for the application, while the SUNFLOOR tool designs a custom topology. For designing ASSoCs that support multiple applications, we apply the extended synthesis procedure presented in Chapter 6. When the design is a CMP that runs software tasks, we apply the synthesis

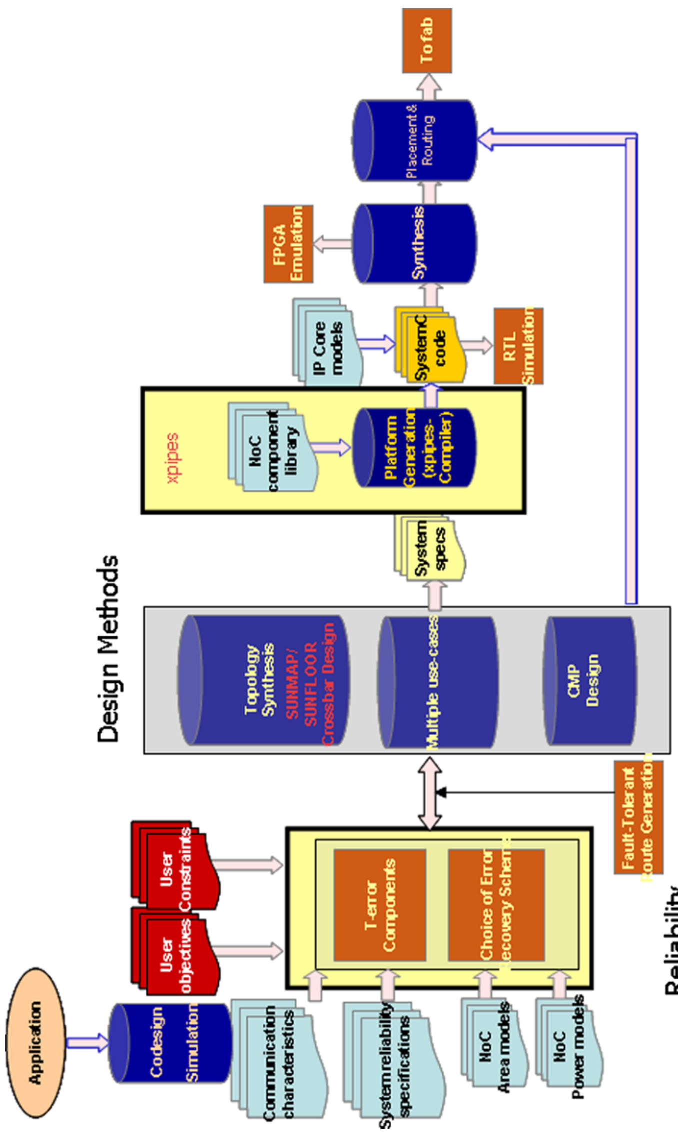

Fig. 12.1 Reliability enhanced Netchip design flow

approach presented in Chapter 7. The individual crossbar switches of the NoC can be further optimized using the method presented in Chapter 2.

When choosing paths for traffic flows, we determine whether it is efficient to apply the multipath routing presented in Chapter 10. We also determine the fault-tolerance level achievable using such a method. We iterate between the error recovery methods chosen in the previous steps with the multipath method, until a design with least area-power overhead that still satisfies the reliability constraints is obtained.

Once a NoC topology that satisfies the reliability constraints is obtained, we proceed to generate the RTL design of the NoC components, as presented in Chapter 3. For this, we use ×pipes, a library of SystemC soft macros for the network components, and the associated tool ×pipesCompiler to generate the entire design. We proceed with the RTL simulation, synthesis, emulation and layout of the design using standard tool chains. This automates some of the most critical and time intensive NoC design steps such as topology synthesis, core mapping, crossbar sizing, route generation, resource reservation, achieving fault-tolerance, RTL code, and layout generation.