9. Future Extensions

In this chapter, some possible future extensions for the Xplorer project are discussed. According to the division of the framework, the extension suggestions are identified for the following three parts of the Xplorer:

- extensions for the KressArray design space
- extensions for the multi-architecture datapath synthesis system
- extensions for the design space exploration system

The issues suggested here are a selection of possible extensions and can therefore not claim to be complete. Furthermore, the realization of these extension may arise problems, which are outlined in the according description.

Extensions to the KressArray Design Space

The extension projected here applies to the definition of the KressArray architecture family, thus extending the design space.

Extended Array Topologies. Although the selected mesh-based approach covers most application requirements and resembles a good generalization of existing structures for reconfigurable architectures, there are applications, which require other topologies. For example, certain systolic algorithms for matrix multiplication [Kung82] are based on a hexagonal arrangement of PEs. However, extending the design space to such exotic topologies implies problems with the VLSI implementation of the array, as non-rectangular cells for wiring by abutment are yet uncommon in VLSI design.

Extended Connection Topologies. In addition to nearest neighbor connections and backbuses, which are arranged horizontally or vertically, a number of other connection topologies can be found in existing architectures, which could be included in the KressArray design space. For example, the MATRIX architecture (see section 2.1.4) features nearest neighbor connections of length two as well as diagonal connections. Such communication resources could enhance the design space, allowing for better-suiting architectures. But they would also imply a greater complexity to handle for the MA-DPSS mapper, which would result in longer times for the synthesis step. This raise of complexity would probably also be encountered for the design suggestion generation process.

Extensions to the MA-DPSS

The suggestions in this subsection apply to the MA-DPSS subsystem of the Xplorer framework, which is used for both generation of status evaluation and production of final mappings.

Alternative Input Languages. While the ALE-X language is sufficient to describe most real-world datapaths, several important features of the KressArray design space can not be expressed, like cycles in datapaths or port specifications. While ALE-X may still be retained to provide compatibility with higher level design frameworks like the CoDe-X environment [Beck97], an alternative language approach appears reasonable. This high level language could allow for the use of all features of the KressArray design space, without relying on manipulation of the α-intermediate format. However, the development of such a language seems non-trivial when looking at advanced features like double output operators. Here, a totally new syntax concept would have to be developed, which would probably lead to acceptance problems for the language.

HDL Model Generator. The MA-DPSS scheduler produces performance estimations for a mapping of an application. However, for simulation and performance estimation systems with embedded KressArrays, the behavior of the application mapped onto the array could be described in a HDL model, which would serve this purpose. Such a model could be generated automatically from the mapping and the architecture information by an according tool.

Library Management. To extend the application of the MA-DPSS under the aspect of a synthesis system, a powerful set of libraries and an according library management tool would improve the MA-DPSS. Such a tool would enhance the realization of a library concept, which is already supported by the mapper. There are two ways of implementing such a concept. One described in section 6.4.3.3 includes the library elements after the mapping is done, which does not need to reveal the datapath of the library element. A tool implementing this approach would need to provide a convenient way of interface definition and floorplanning of the array layout. The second approach consists in the inclusion of the library datapath into the application datapath, with the resulting total datapath being subject to the mapping step. Thus, this approach performs a kind of macro expansion.

Extensions to the Exploration Framework

The enhancements envisioned in this subsection apply to the general exploration approach or deal with additional tools, which improve the usability of the Xplorer framework.

Instruction Set Mapper. In order to explore different operator sets, the user has to adapt the datapath manually. This process could be done automatically by an according instruction set mapper tool. The task of this tool consists in replacing operators or groups of operators, which implement a certain function, by an alternative implementation of this function, which may again comprise one or more operators. In the case of the replacement of one operator by a sub-tree of several ones, the task resembles a macro expansion. The tool would thus be adequate for implementing the second approach for a library concept discussed above. If the source function consists of several operators, a tree matching algorithm could be employed. However, the instruction set mapper tool would also have to implement a usable management for different operator sets. Such operator sets would typically represented by a specification of those functions, which are subject to possible replacement, and a set of implementations for those functions.

Chip Area Estimation. A possible extension for the analyzer subsystem consists in a chip area estimation plug-in, which could be used to realize constraints for the architecture. This plug-in would have to consider the operator set as well as the communication resources of the architecture in order to produce useful estimations. Thus, the exploration process could be extended to lower levels of abstractions. However, an area model depends highly on the implementation of the resources, including different types of ALUs, in case of multiple operator sets. Thus, the realization of such a tool appears not trivial.

Automatic Rule Generation. In order to extend the suggestion generation features of the Xplorer framework, the user has to create the according fuzzy rules based on his or her expert knowledge or on experiments conducted. As described in section 8.4, the generation step for one suggestion resembles the process flow of a fuzzy controller. There exist several approaches for the automatic generation of rules for such controllers. For example, in [LiLi95] a system is introduced, which is capable of learning fuzzy rules during the control process by an additional feedback. Another work in [Bona96] uses a genetic algorithm to develop rules automatically. An overview on other approaches can also be found in [Bona96]. However, the fuzzy rule sets employed in the Xplorer framework may use a relative high number of input variables. Thus, the automatic generation of rules for the Xplorer would most probably include a very high complexity, rendering this extension an ambitious goal.