This chapter focuses on the ISE generation algorithms\(^1\) which form the backbone of our ISA customization framework. Currently, two different ISE extraction algorithms – one based on integer linear programming and the other on high level synthesis – have been integrated into our design flow. Both of these algorithms will be discussed in detail in the subsequent sections.

Before going into the details of the individual ISE generation algorithms, Sect. 8.1 provides a precise mathematical formulation of the ISA customization problem. Some of the notations and assumptions used in this formulation have already been introduced in Chaps. 6 and 7, but they are re-stated here for the sake of completeness and convenience.

### 8.1 Mathematical Formulation of the ISA Customization Problem

As has been already mentioned in Chaps. 6 and 7, the goal of an ISE generation algorithm is to extract a set of special instructions from a target application to maximize its execution performance. Each special instruction corresponds to a cluster of operations from an application’s CDFG and is implemented in a custom functional unit inside an ASIP base processor architecture. Any cluster of operations selected as a special instruction must obey the generic, architectural and CFU interface constraints described in Sect. 6.2.

\(^1\)For the sake of convenience, ISE generation algorithms will be interchangeably called ISE extraction algorithms, ISE identification algorithms or simply, generation algorithms for the rest of this book.
An ISA generation algorithm is provided with the following three kinds of inputs:

1. **Hot-spots of the target application** which are to be accelerated using ISEs. Application hot-spots correspond to the most computationally expensive segments of code (i.e. functions, basic blocks or loop kernels). Since the ISE generation algorithms operate on the CDFG representation of a given application, hot-spots need to be defined in terms of CDFG elements.

   The CDFG of an application consists of a set of basic blocks, \( B = \{ b_1, b_2, \ldots, b_k \} \), connected by control-flow edges. The body of each basic block can be represented by a DFG, \( G = (V, E) \), where each node represents either an operation, or a variable, or a literal constant, and the edges represent the true data-dependencies (or, flow dependencies) between the nodes. The node set, \( V \), of any basic block is composed of the following three mutually non-overlapping and possibly empty subsets

   (a) \( V_{\text{NON-OP}} \) which contains the variable and constant nodes in \( V \).
   (b) \( V_{\text{CAND}} \) which contains the set of CANDidate nodes in \( V \) that are eligible for inclusion in a special instruction.
   (c) \( V_{\text{FORBID}} \) which contains operations that are FORBIDden inside an ISE, and must be implemented in the base processor core. Nodes belonging to \( V_{\text{FORBID}} \) are called forbidden nodes, and may include (but are not limited to) function calls, jumps and floating point operations.

   Using the CDFG terminology, application hot-spots supplied as inputs to an ISE generation algorithm can be defined as a set, \( B_{HS} \subseteq B \), which contains the most computationally intensive and most frequently executed basic blocks. A generation algorithm usually iterates over the elements of \( B_{HS} \) one by one, and extracts ISE definitions from each basic block by clustering nodes from the corresponding \( V_{\text{CAND}} \) set.

2. **A set of architectural constraints** that characterize the CFU interface as well as the size and delay restrictions on ISE data-paths. The following user specified architectural constraints are considered by our ISE generation algorithms.

   (a) \( GPR_{\text{IN}}^{\text{MAX}} \) and \( GPR_{\text{OUT}}^{\text{MAX}} \) which specify the maximum number of input and output operands accessible from the GPR file to a special instruction.
   (b) \( MEM_{\text{READ}}^{\text{MAX}} \) and \( MEM_{\text{WRITE}}^{\text{MAX}} \) which specify the maximum number of main memory read and write operations permissible from an ISE.
   (c) \( LAT_{\text{MAX}} \) which specifies the maximum number of base processor clock cycles an ISE can take to compute and commit its results.
   (d) \( AREA_{\text{MAX}} \) which specifies the maximum total area of the identified ISEs.

   This constraint can be alternatively specified by placing a restriction on the maximum number of arithmetic/logical/comparison computational resources in the CFU.

3. **A set of architectural parameters** to guide the identification heuristics. Especially, these parameters are used for area and latency estimation of candidate ISEs. Each operation node, \( v_i \in V \), for a basic block, \( b \in B_{HS} \), is associated...