The goal of a system-level design methodology is to decrease design cost and design time. Firstly, the complexity of a modern system does not allow us to describe its implementations directly. Furthermore, it is difficult to create derivative implementations with different functions or different architectures, because functions and architectures cannot be extracted easily from implementations for reuse. Therefore a separation of function, architecture and implementation is necessary when designing a system. Design activities are needed to combine different functions and architectures for an implementation to decrease design time and design cost.

To be able to separate function, architecture and implementation in a system design flow, different abstraction levels need to be defined. The idea is to gradually confront designers with implementation details such as timing and data representations. Abstraction levels are used to describe the functionality of the target design by mathematical equations and/or algorithms. It then goes through behavior synthesis processes to generate register transfer level (RTL) circuit (Fig. 3.1).

![System level design](image)

Fig. 3.1 System level design
3.1 Abstraction Levels

A design can be described on different levels of abstraction. Raise the abstraction level is always a trade-off between the speed and accuracy of a potential simulation model. Function level only captures the algorithm regardless of the implementation details, an algorithmic model has a huge advantage in its high simulation speed. On the other end, RTL simulation accuracy is fidelity to real implementation. But it is too expensive to pay due to its lengthy simulation time.

3.1.1 Algorithm Level

The motivation for introducing this algorithmic level of abstraction is to quickly obtain a function to determine what the system is supposed to do, without making architecture assumptions. Hence there is the potential to reuse functions either to create derivative functions, or to synthesize different implementations with different architectures.

Algorithm Level contains two main subjects: algorithm specification and data communication.

In the algorithm specification, an executable functional specification of the algorithm is created. It may be a C/C++ or Matlab code. This executable specification is used to check the validity of the algorithm. The simulation in this design step is sequential, which has no timing information and uses a single thread of control. The simulation speed is high due to lack of timing and architecture details.

Profiling techniques are used to obtain an initial estimate of the computational load of the different functions and the amount of data transfer between them.

Code inspection is used to estimate the amount of flexibility required for each of the functions. The results of both, code inspection and profiling, are used as input for a task and, in a later stage, for hardware/software partitioning.

By the executable algorithm specification, a golden reference model is generated for verification throughout the whole flow.

With the design constraints and requirements and a suitable architecture template in mind and the results from the previous algorithm design step, the system is partitioned into tasks that perform functions and channels through which data are communicated between these tasks. With a multi-threaded simulation tool, the communication load on the channels and the computation load on the tasks can be analyzed. If necessary, the system can be repartitioned to meet the constraints and requirements.

3.1.2 Architecture Level

The motivation of this architecture level is to quickly find an efficient architecture implementation. Efficiency can be defined in terms of power, timing, area, etc. To be able to quickly evaluate the efficiency of alternative implementations, we want