Chapter 13. Overview of Parallel Code Generation

Authors: *Soonhoi Ha*

This chapter describes the overall procedure of parallel code generation in Ptolemy. We start with an SDF program graph and a multiprocessor target description. In the target definition, we specify the number of processors and some information about the processors with target parameters. If the number of processor is given 1, it is classified as a sequential code generation problem: a chosen SDF scheduler schedules the graph and code is generated based on the scheduling result. Parallel code generation is a bit more complicated.

If the number of processor is greater than 1, we create an APEG (acyclic precedence expanded graph) associated with the SDF program graph. The APEG graph displays all precedence relations between invocations of the SDF stars. All parallel schedulers take this APEG graph as an input graph and generate the schedule. In Ptolemy, we can have many scheduling algorithms (currently 3), and choose one by setting the appropriate target parameters. There is a common framework all parallel scheduling algorithm should be fit into . The scheduling result indicates the assignment and the ordering of star invocations in the processors. The next step is to generate code for each processor based on the scheduling result.

We create an SDF *sub-universe* for each processor. The sub-universe consists of stars assigned to the processor and some other automatically inserted stars, for example send and receive stars for interprocessor communication. We apply the sequential code generation routine for each processor with the associated sub-universe.

We may generate parallel code inside a wormhole so that the main workstation can communicate with the target multiprocessor system. Then, the wormhole interface code should be added to the generated code.

The following chapter will explain each steps in significant detail with code segments.