Chapter 7. SDF Domain

Authors: *Joseph T. Buck*
Soonhoi Ha
Edward A. Lee

7.1 Introduction

Synchronous dataflow (SDF) is a statically scheduled dataflow domain in Ptolemy. “Statically scheduled” means that the firing order of the stars is determined once, during the start-up phase. The firing order will be periodic. The SDF domain in Ptolemy is one of the most mature, with a large library of stars and demo programs. It is a simulation domain, but the model of computation is the same as that used in most of the code generation domains. A number of different schedulers, including parallelizing schedulers, have been developed for this model of computation.

We assume in this very short chapter that the reader is familiar with the SDF model of computation. Refer to the *User’s Manual*. Moreover, we assume the reader is familiar with chapter 2, “Writing Stars for Simulation”. Since most of the examples given in that chapter are from the SDF domain, there is only a little more information to add here.

7.2 Setting SDF porthole parameters

All stars in the SDF domain must follow the basic SDF principle: the number of particles consumed or produced on any porthole does not change while the simulation runs. These numbers are given for each porthole as part of the star definition. Most stars consume just one particle on each input and produce just one particle on each output. In these cases, no special action is required, since the porthole SDF parameters will be set to unity by default. However, if the numbers differ from unity, the star definition must reflect this. For example, the `FFTCx` star has a *size* parameter that specifies how many input samples to read. The value of that parameter specifies the number of samples required at the input in order for the star to fire. The following line in the `setup` method of the star is used to make this information available to the scheduler:

```
input.setSDFParams (int(size), int(size)-1);
```

The name of the input porthole is *input*. The first argument to `setSDFParams` specifies how many samples are consumed by the star when it fires; it is the same as the number of samples required in order to enable the star. The second argument to `setSDFParams` specifies how many past samples (before the most recent one) will be accessed by the star when it fires.

If the number of particles produced or consumed is a constant independent of any states, then it may be declared right along with the declaration of the input, in the `.pl` file. For example,

```
input {
     name { signalIn }
     type { complex }
```

```
                numTokens { 2 }
                desc { Complex input that consumes 2 input particles. }
        }
```

This declares an input that consumes two successive complex particles.