Key Particle methods

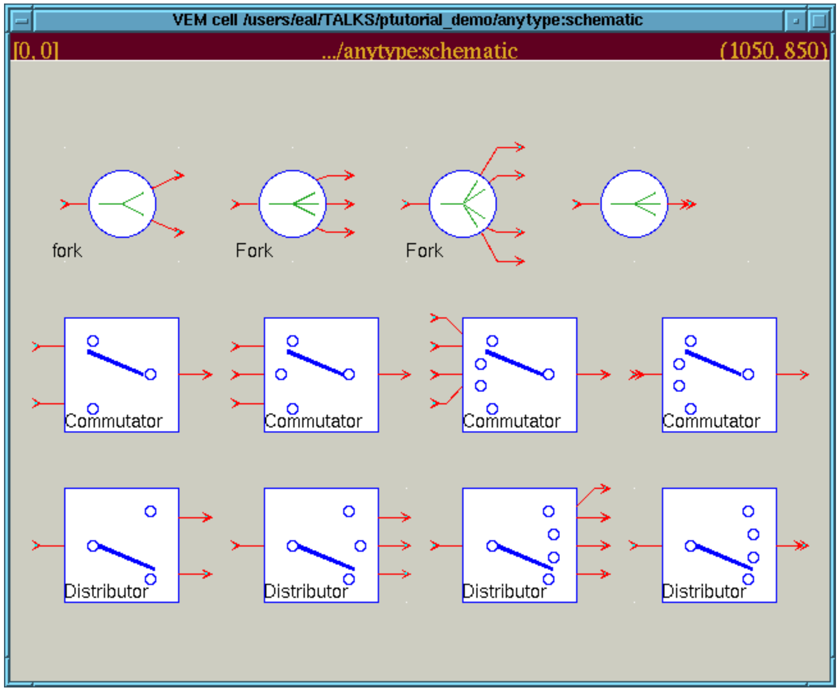
```
class Particle {
public:
    virtual operator int ();
    virtual operator float ();
    virtual operator double ();
    ... etc...

    virtual StringList print ();
    virtual Particle& initialize();

    // Load the Particle with data
    virtual void operator << (int);
    virtual void operator << (double);
    ... etc...

    // Copy a Particle
    virtual Particle* clone();

    // compare two particles
    virtual int operator == (const Particle&);
```

casts to standard types convert particles to data and also change types.

generate ASCII

reset to "zero"

Anytype



Fork:
replicate a particle

Commutator:
merge particle streams

Distributor:
split particle streams

Red stems indicate stars that operate on any particle. The code inside refers only to type “Particle,” so these stars are polymorphic.

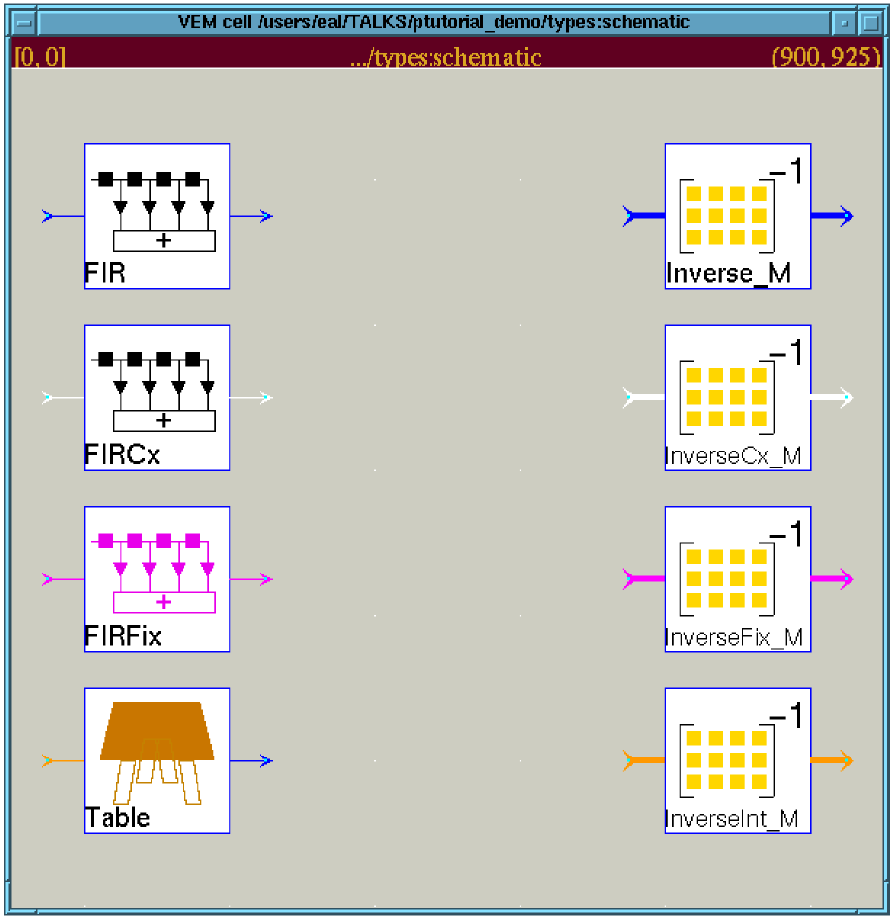
Color Hints at Data Types

blue = float (double)

white = complex

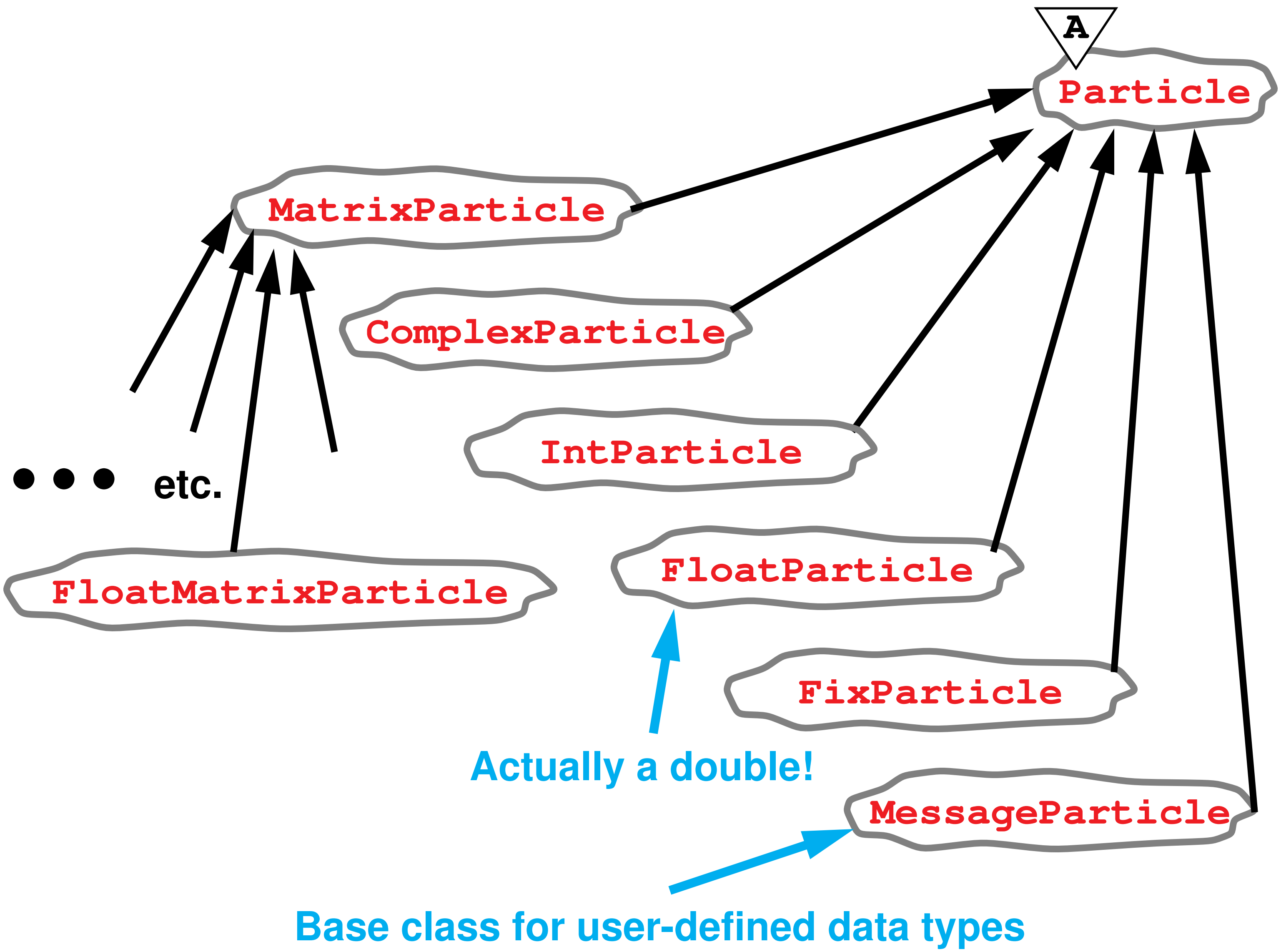
magenta = fixed point

orange = integer

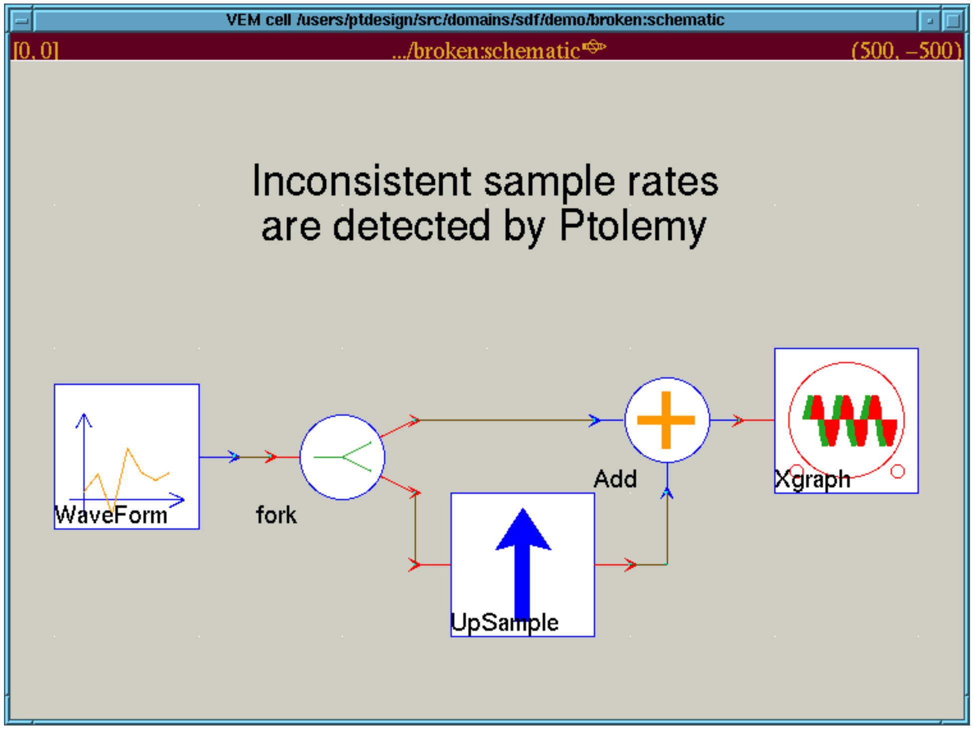


thicker stems indicate matrices

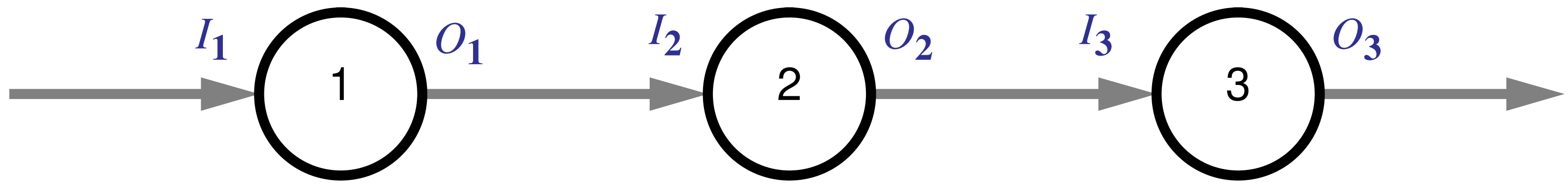
Data Types in Ptolemy



Example of an Inconsistent System



Consistency



Balance equations:

$$r_1 O_1 = r_2 I_2$$

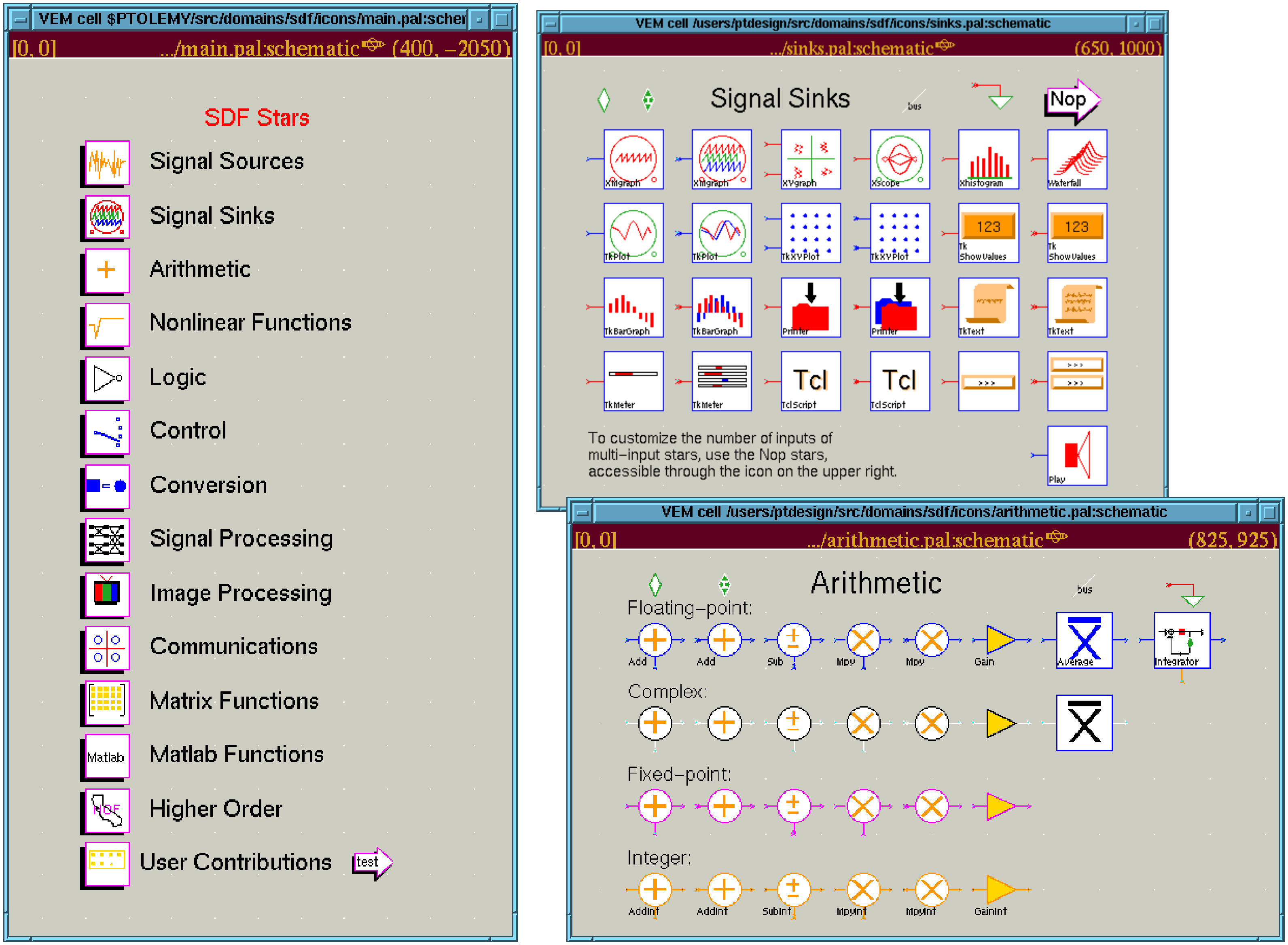
$$r_2 O_2 = r_3 I_3$$

Solve for the smallest integers r_i.

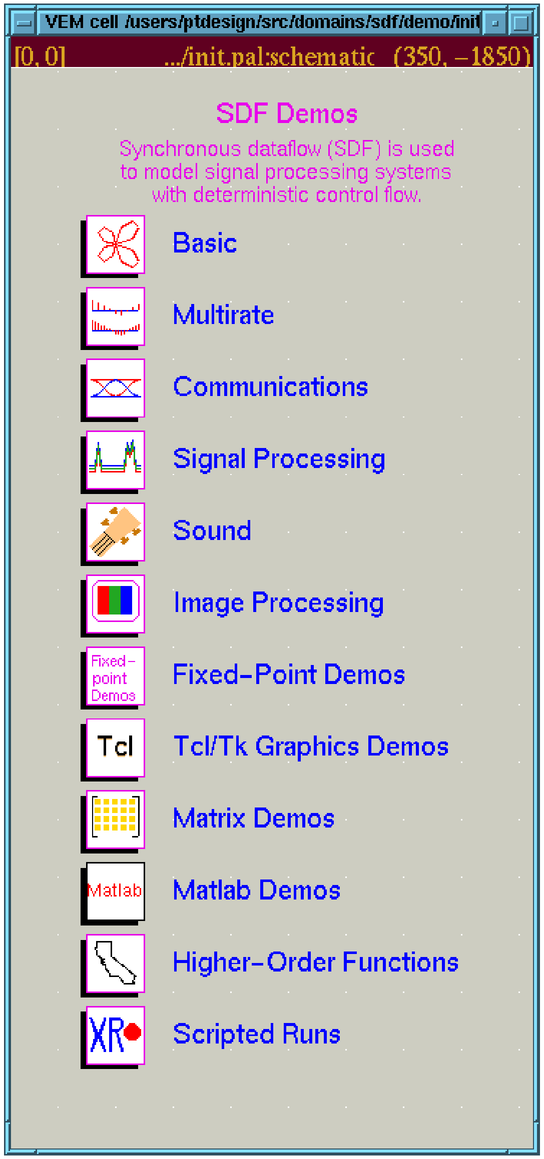
Then schedule according to data dependencies until repetitions r_i have been met for all actors.

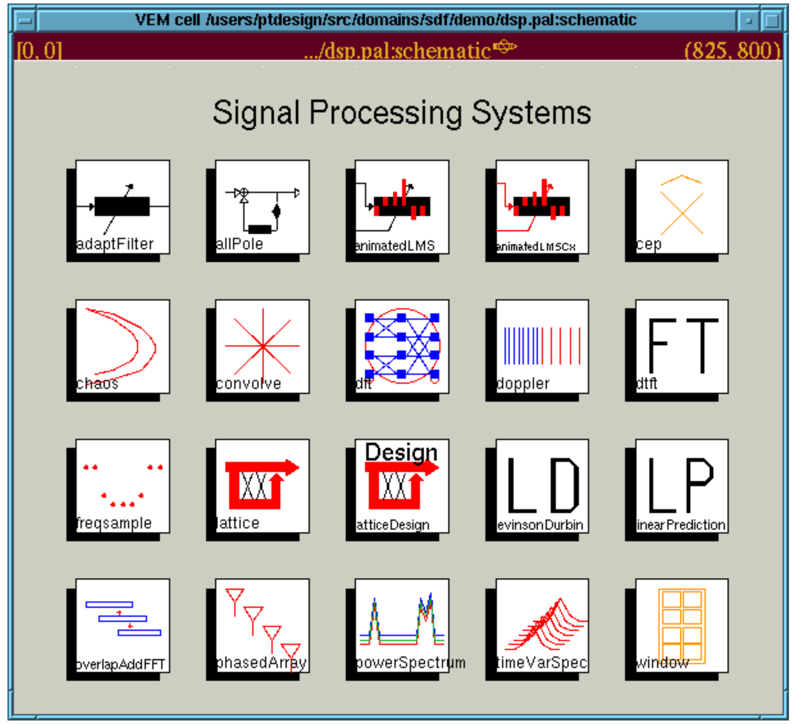
The balance equations have no solution if the graph is *inconsistent*.

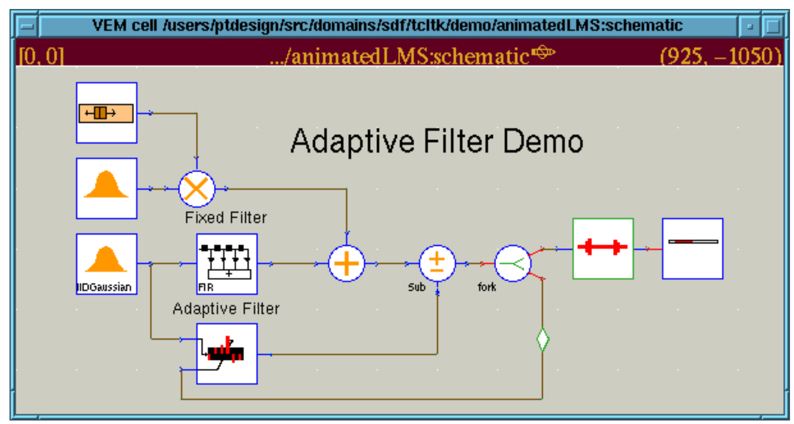
Some SDF Stars



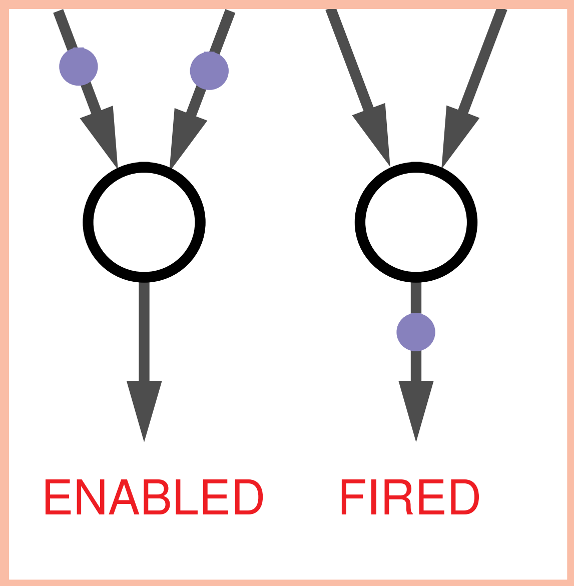
Some SDF Demos







Synchronous Dataflow



Properties

- Flow of control is predictable at compile time
- Schedule can be constructed once and repeatedly executed
- Suitable for synchronous multirate signal processing