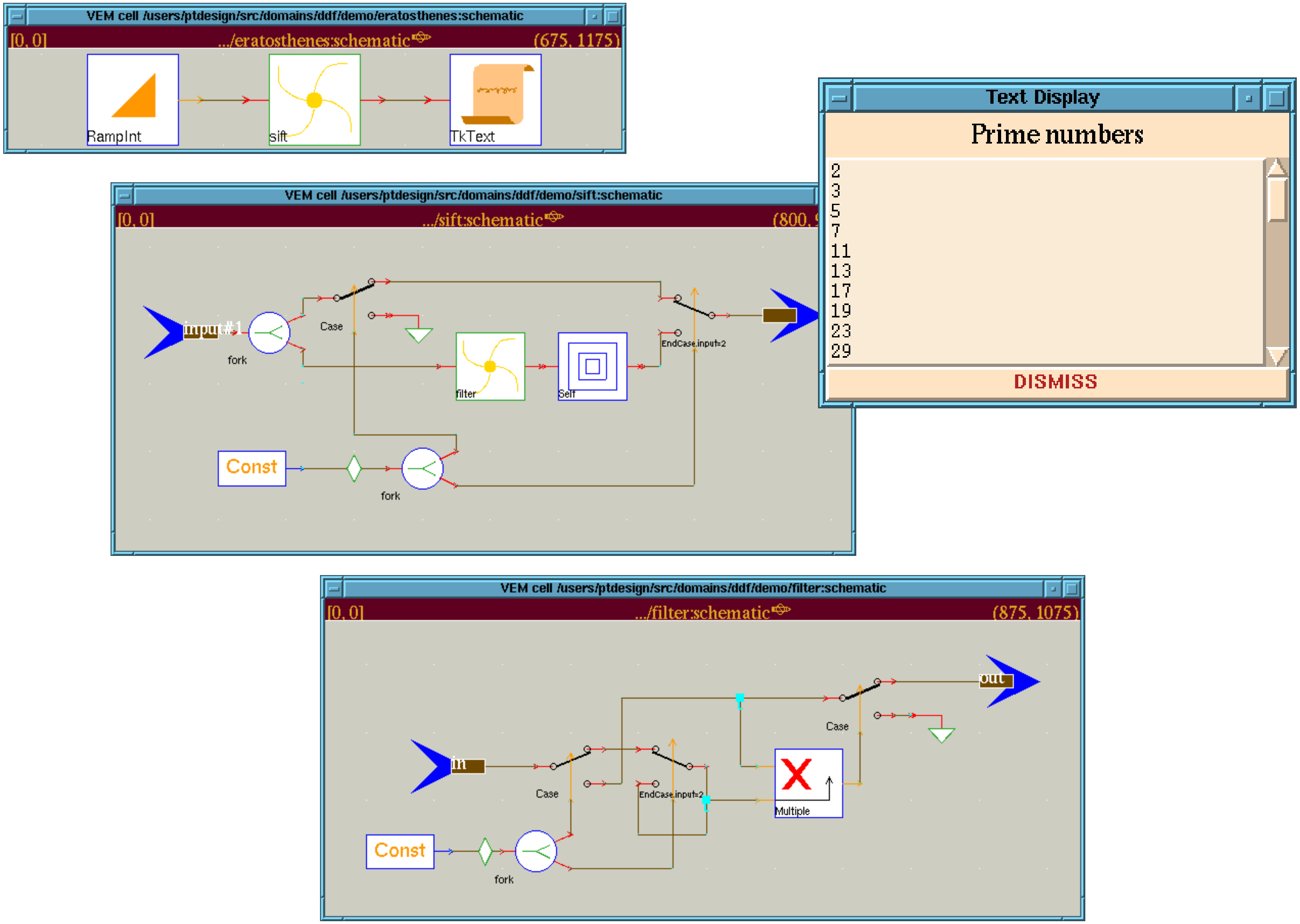
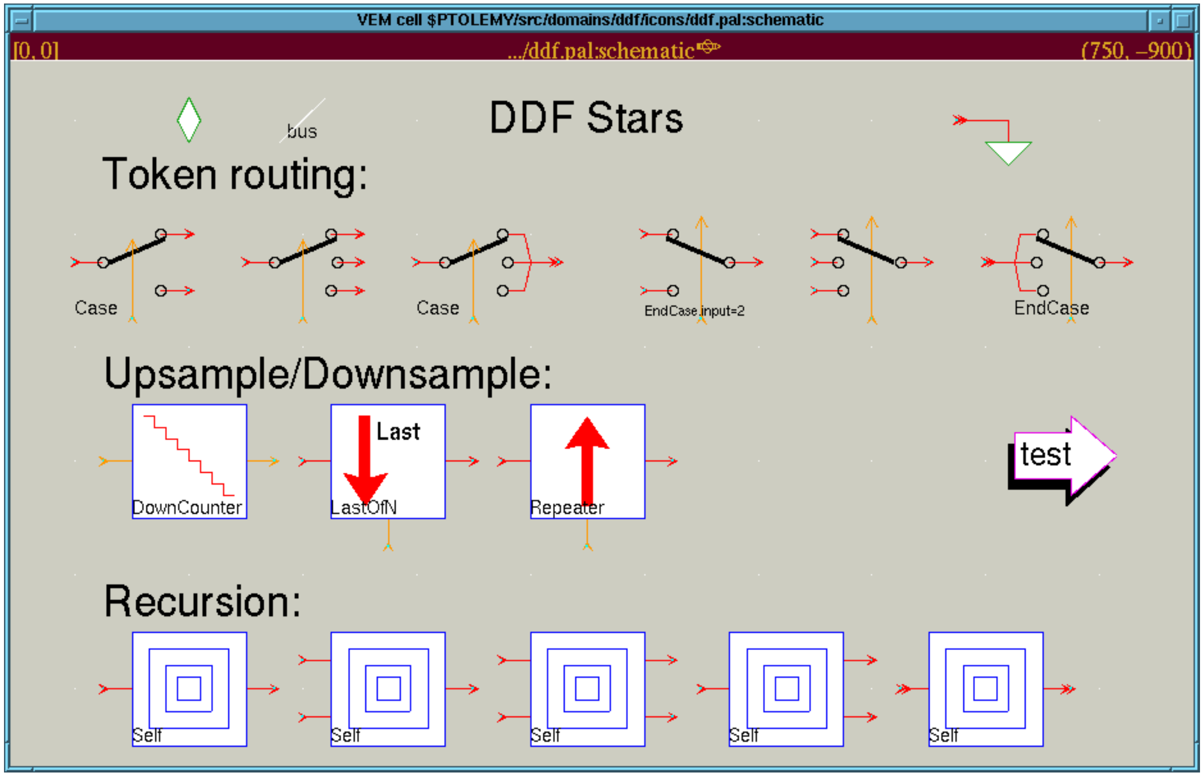
A Demo in the DDF Domain



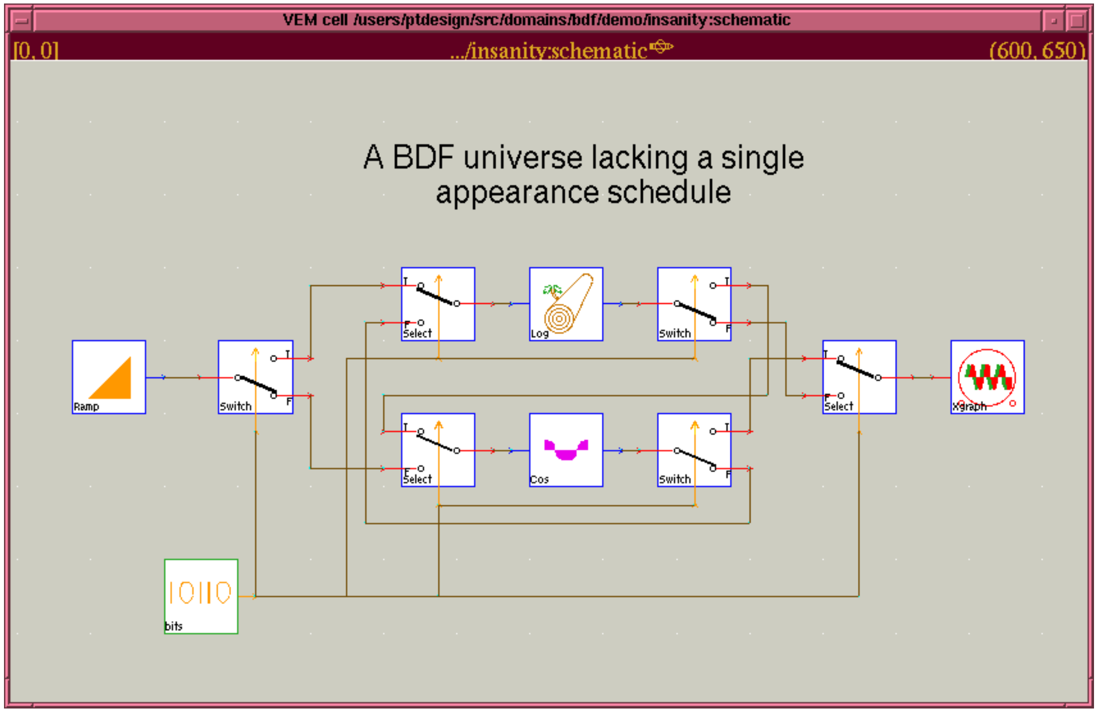
The Dynamic Dataflow Domain



The schedulers in this domain make no attempt to construct a compile-time schedule.

Note that the notion of an iteration, and the "when to stop" control, may not be what you expect.

When the BDF Scheduler Gives Up



Dynamic execution of 4 clusters:

Cluster 1:
- Select1
- Log1
- Switch1

Cluster 2:
- Select2
- Cos1
- Switch2

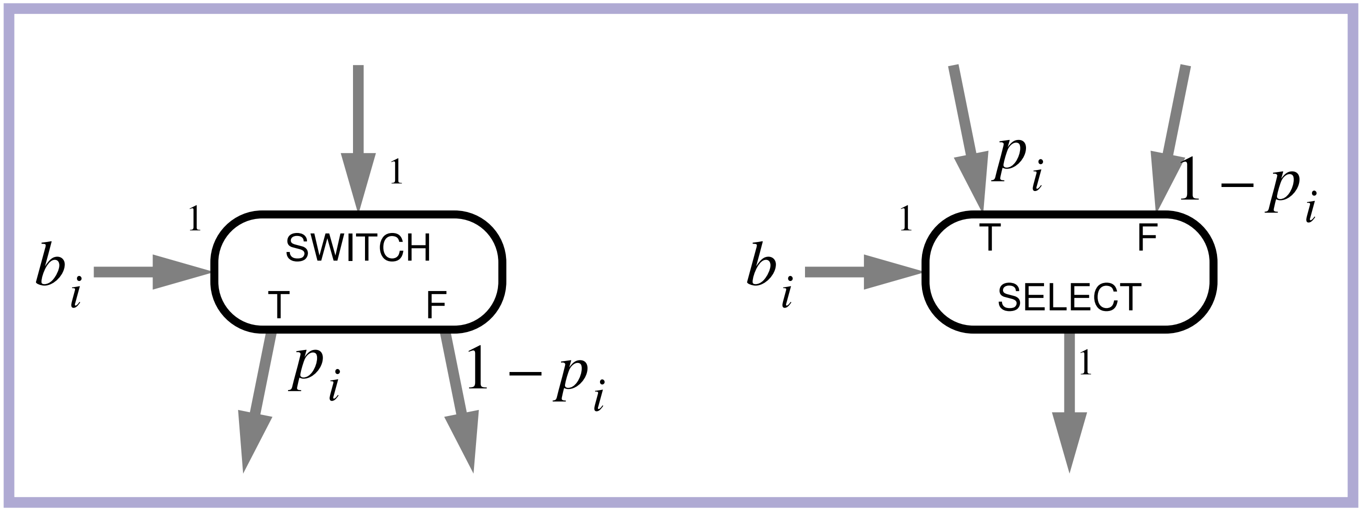
Cluster 3:
- Select3
- Xgraph1

Cluster 4:
- bits1
- auto-fork-node1
- Ramp1
- Switch3

This system randomly chooses between computing cos(log(x)) and log(cos(x)).

The BDF scheduler fails to find a static schedule for this system, despite the fact that one exists.

Balance Equations Generalize to Boolean Dataflow



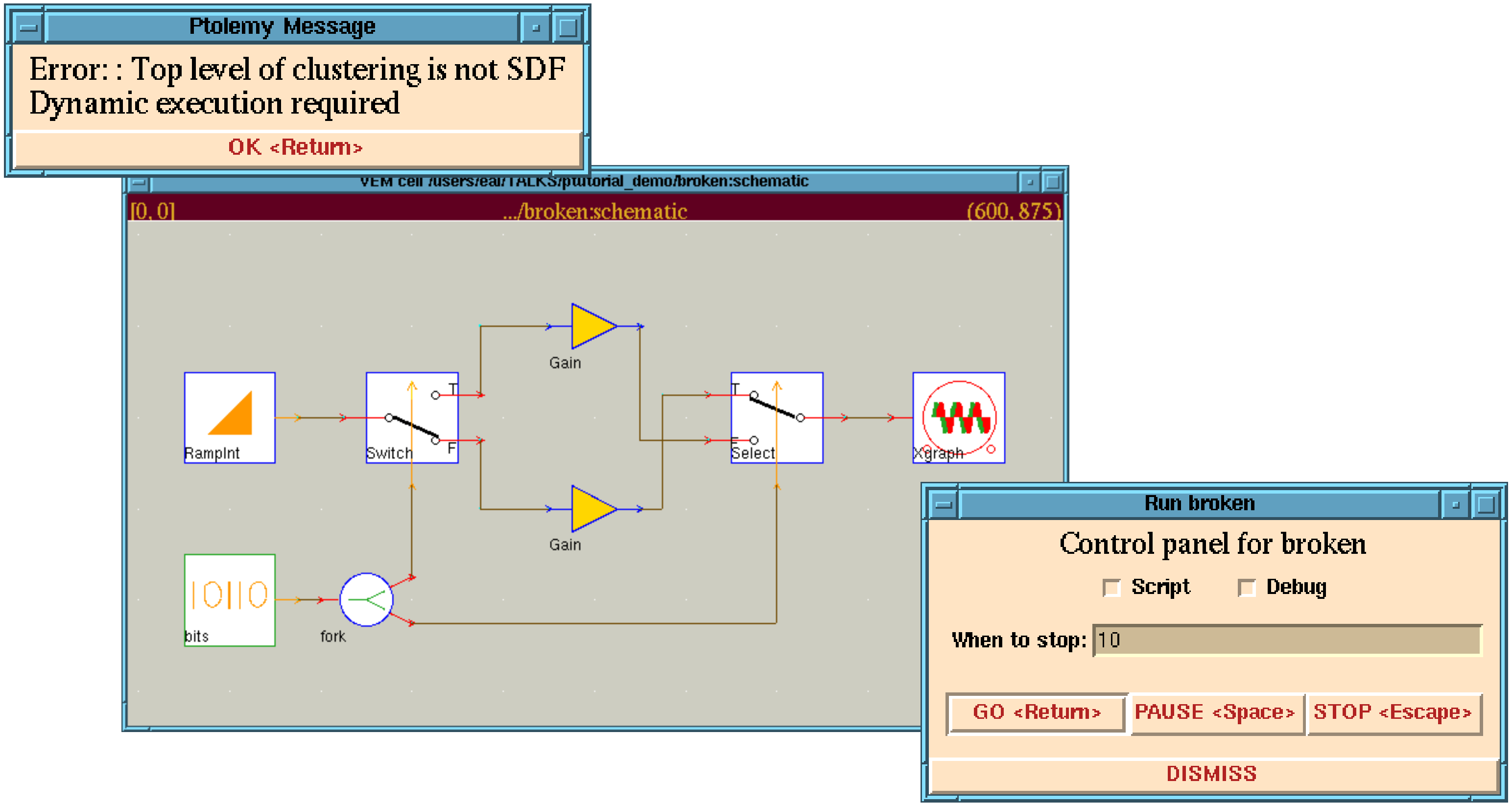
The variable p_i is a symbolic placeholder for an unknown.

The balance equations are solved symbolically.

However ...

Verification and scheduling problems (deadlock, bounded memory) become undecidable. No scheduler can always find a compile-time schedule for BDF graphs, even when one exists.

Inconsistent System in BDF

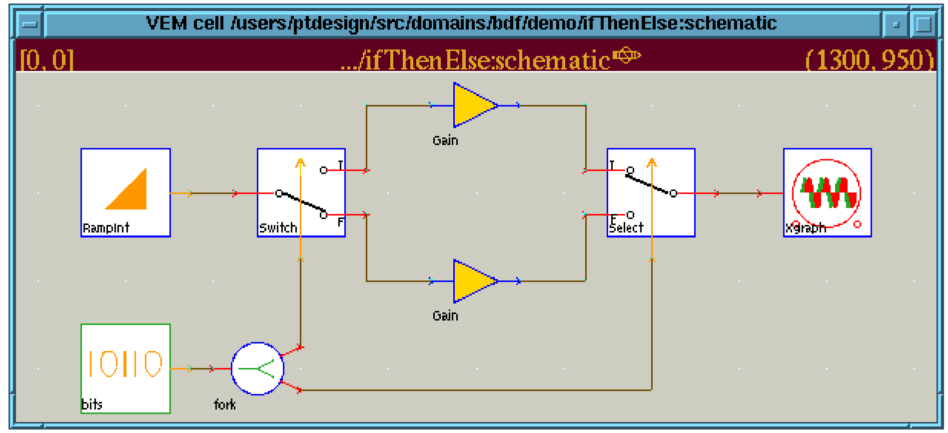


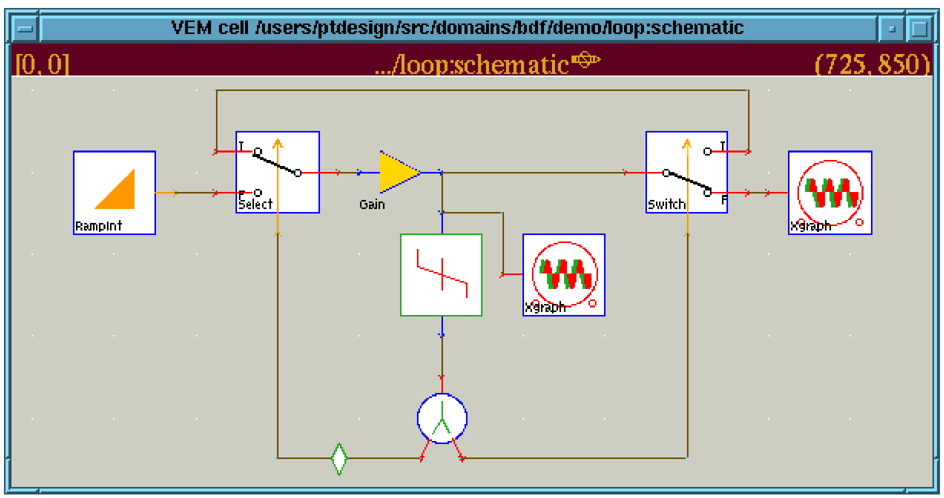
Change the Target to default-DDF to execute the sysetm.

However, the “when to stop” argument in the control panel is meaningless. How to properly control such dynamic executions remains a research problem.

Some BDF Demos and their Schedules

```
bits1
Fork1
RampInt1
Switch1
if(Fork1_output#2) Gain1
if(!Fork1_output#2) Gain2
Select1
Xgraph1
```






```
RampInt1
do {
    Select1
    Gain1
    auto-fork
    Xgraph2
    Thresh1
    Fork1
    token := <value from
        Fork1_output#2>
    Switch1
} while (token)
Xgraph1
```

Boolean Dataflow (BDF)