The Synchronous Data Flow Domain in Ptolemy II

Steve Neuendorffer
UC Berkeley

SDF Overview

- Useful for simple Dataflow systems without control
  - Signal Processing
  - DSP code synthesis
- Enables efficient execution
  - No threads!
  - No deadlock detection!
  - No time ordered queues!
**SDF Requirements**

- Data rates on each port are constant and known a priori
  - unspecified rates are assumed to be one
- Delays are added to avoid deadlock
  - represented by initial tokens
- Graph is consistent (a static schedule exists)

**Example SDF graphs**

- **Consistent graph**
  - A Valid Schedule: ABBC
  - This schedule is also the minimum valid schedule.

- **Inconsistent graph**
  - No valid schedule exists!
  - Tokens will always accumulate between C and A
**SDF Operation**

- Scheduler creates a minimum valid schedule, if one exists.
  - Buffers on relations have bounded size
  - Each actor fires the minimum number of times necessary.
- Schedule is executed an integral number of times for each firing of the system
- Only linear (Single Processor) schedules are currently supported.

**Vector Quantization Demo**

- 8 Dimension VQ
- Each image is separated into 4x2 blocks, and the best approximation to the block is chosen from a codebook.
- Fast implementation using table lookups
- Video compression at 8:1
Vector Quantization Demo

Sequence \(\rightarrow\) Partition \(\rightarrow\) HTVQEncode

Original \(\rightarrow\) Display

Unpartition \(\rightarrow\) VQDecode

Display

Compressed

IntMatrix

3168

4x2 IntMatrix

176x144 IntMatrix

IntMatrix

Original

Compressed

Display

HTVQEncode

VQDecode

Original

Compressed

Display

IntMatrix

3168

4x2 IntMatrix

176x144 IntMatrix

Original

Compressed

Display

IntMatrix

3168

4x2 IntMatrix

176x144 IntMatrix

Original

Compressed

Display

IntMatrix

3168

4x2 IntMatrix

176x144 IntMatrix

Original

Compressed

Display

IntMatrix

3168

4x2 IntMatrix

176x144 IntMatrix

Original

Compressed

Display

IntMatrix

3168

4x2 IntMatrix

176x144 IntMatrix

Original

Compressed

Display
Expression Demo

- Based on Expression Actor
  - Type Polymorphic
    - Works with any ScalarToken
  - Domain Polymorphic
  - Shape Polymorphic
    - Operation depends on number and type of ports
  - Performs arbitrary arithmetic on input scalar tokens

```
Expression Demo

init=0.0
step=0.1*PI

Expression
PlotBox

init=0.0
step=0.01*PI

init=0.0
step=0.1*PI
```
Expression Demo

Communications System Demo

- Simulates a baseband signaling communications system operating under Additive White Gaussian Noise channel
- Raised Cosine pulse shape
  - Implemented using standard FIR filter actor.
Communications System Demo

CoinFlip → LineCoder → RaisedCosine + Gaussian

TimePlot ← RaisedCosine

In addition:
- Pulse Shaping filter
- Inverse Matched filter

Eye Diagram
SDF Summary

- SDF allows many signal processing algorithms to be modeled with very little runtime overhead
- Very similar to SDF domain in Ptolemy Zero, but currently only supports simple schedules.