



Concurrent Models of Computation for Embedded Software



Edward A. Lee

Professor, UC Berkeley
EECS 290n – Advanced Topics in Systems Theory
Fall, 2004

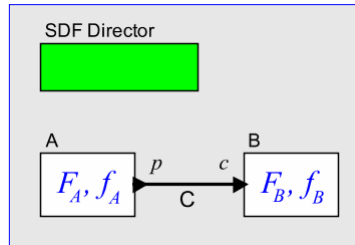
Copyright © 2004, Edward A. Lee, All rights reserved

Lecture 17: Generalizations of SSDF

History of Dataflow Models of Computation

- Computation graphs [Karp & Miller - 1966]
- Process networks [Kahn - 1974]
- Static dataflow [Dennis - 1974]
- Dynamic dataflow [Arvind, 1981]
- K-bounded loops [Culler, 1986]
- Synchronous dataflow [Lee & Messerschmitt, 1986]
- Structured dataflow [Kodosky, 1986]
- PGM: Processing Graph Method [Kaplan, 1987]
- Synchronous languages [Lustre, Signal, 1980's]
- Well-behaved dataflow [Gao, 1992]
- Boolean dataflow [Buck and Lee, 1993]
- Multidimensional SDF [Lee, 1993] **today**
- Cyclo-static dataflow [Lauwereins, 1994]
- Integer dataflow [Buck, 1994]
- Bounded dynamic dataflow [Lee and Parks, 1995]
- Heterochronous dataflow [Girault, Lee, & Lee, 1997]
- Parameterized dataflow [Bhattacharya and Bhattacharyya 2001]
- ...

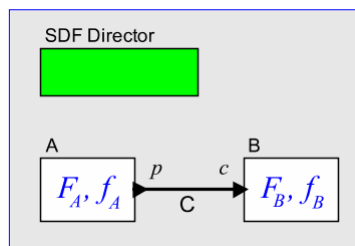
Statically Schedulable Dataflow – SSDF Historically called: Synchronous Dataflow (SDF)



If the number of tokens consumed and produced by the firing of an actor is constant, then static analysis can tell us whether we can schedule the firings to get a useful execution, and if so, then a finite representation of a schedule for such an execution can be created.

Lee 17: 3

Balance Equations



Let q_A, q_B be the number of firings of actors A and B.

Let p_C, c_C be the number of token produced and consumed on a connection C.

Then the system is *in balance* if for all connections C

$$q_A p_C = q_B c_C$$

where A produces tokens on C and B consumes them.

Lee 17: 4

Multidimensional SSDF

(Lee, 1993)

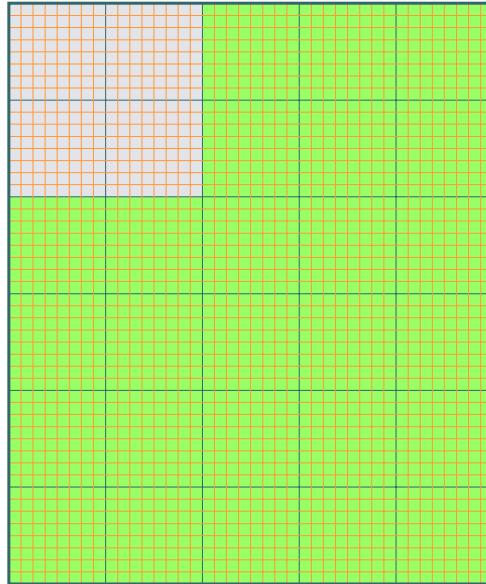
Production and consumption of N -dimensional arrays of data:



Balance equations and scheduling policies generalize.

Much more data parallelism is exposed.

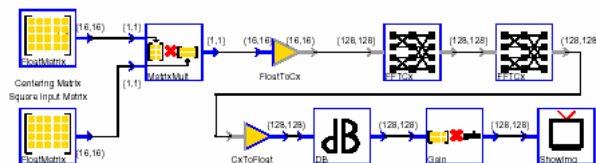
Similar (but dynamic) multidimensional streams have been implemented in Lucid.



Lee 17.5

More interesting Example

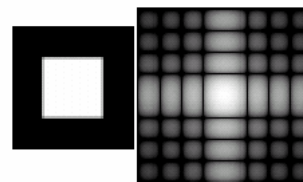
Two dimensional FFT constructed out of one-dimensional actors.



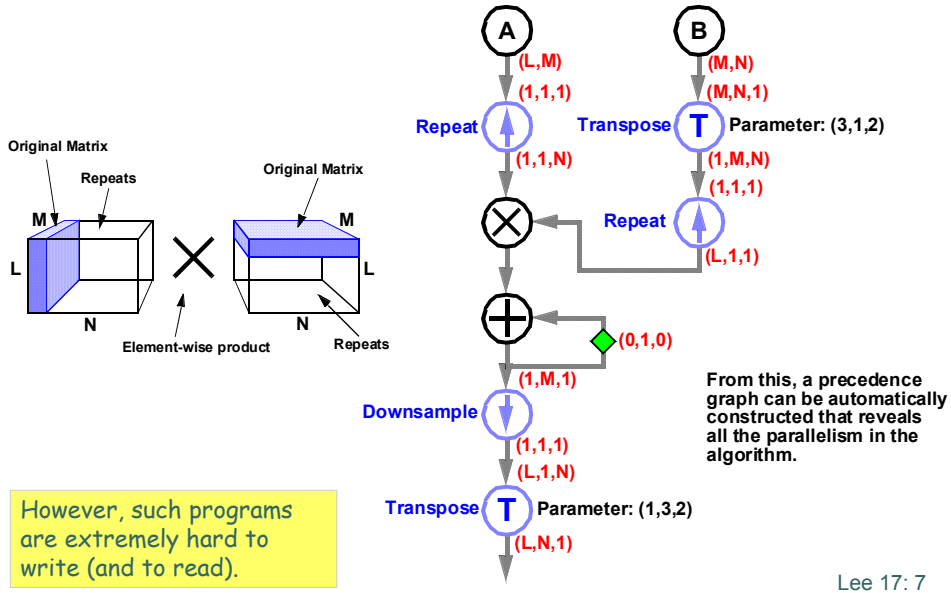
MDSDF SCHEDULE:

```
fft_of_square2.FloatMatrix1, firing range: (0,0)
fft_of_square2.FloatMatrix2, firing range: (0,0)
fft_of_square2.Mult1, firing range: (0,0) - (15,15)
fft_of_square2.FloatToCx1, firing range: (0,0)
fft_of_square2.FFTCx2, firing range: (0,0) - (15,0)
fft_of_square2.FFTCx1, firing range: (0,0) - (0,127)
fft_of_square2.CxToFloat1, firing range: (0,0)
fft_of_square2.DB1, firing range: (0,0)
fft_of_square2.Gain1, firing range: (0,0)
fft_of_square2.ShowImg1, firing range: (0,0)
```

Figure 6. Screen dump of 2D-FFT system, the associated schedule, and outputs.



MDSSDF Structure Exposes Fine-Grain Data Parallelism



Extensions of MDSSDF

Extended to non-rectangular lattices and connections to number theory:

P. K. Murthy, "Scheduling Techniques for Synchronous and Multidimensional Synchronous Dataflow," Technical Memorandum UCB/ERL M96/79, Ph.D. Thesis, EECS Department, University of California, Berkeley, CA 94720, December 1996.

Praveen K. Murthy and Edward A. Lee, "Multidimensional Synchronous Dataflow," *IEEE Transactions on Signal Processing*, volume 50, no. 8, pp. 2064 -2079, July 2002.

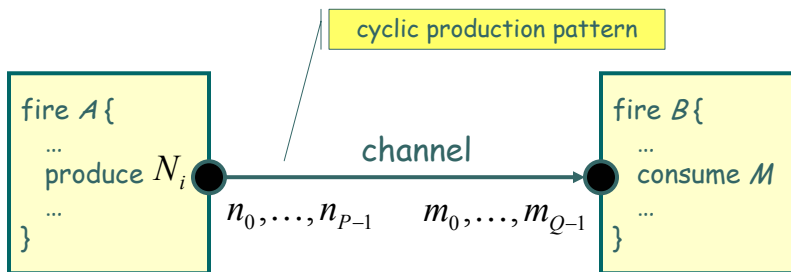
Cyclostatic Dataflow (CSDF)

(Lauwereins et al., TU Leuven, 1994)

Actors cycle through a regular production/consumption pattern.

Balance equations become:

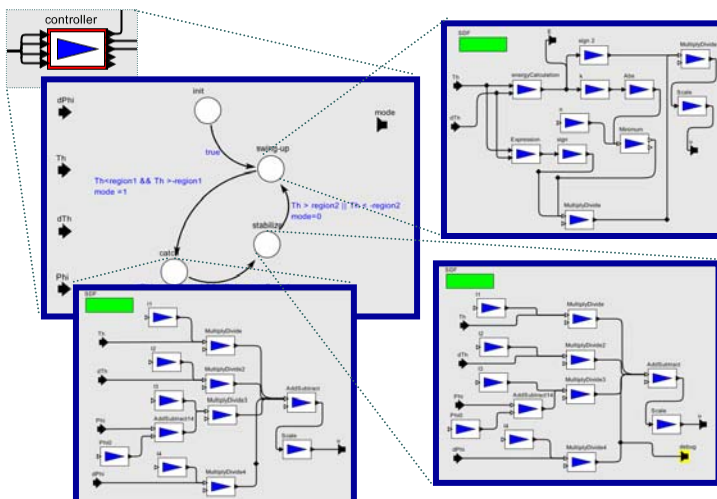
$$q_A \sum_{i=0}^{R-1} n_{i \bmod P} = q_B \sum_{i=0}^{R-1} m_{i \bmod Q}; \quad R = \text{lcm}(P, Q)$$



Lee 17: 9

Heterochronous Dataflow (HDF)

(Girault, Lee, & Lee, 1997)



An actor consists of a state machine and refinements to the states that define behavior.

Lee 17: 10

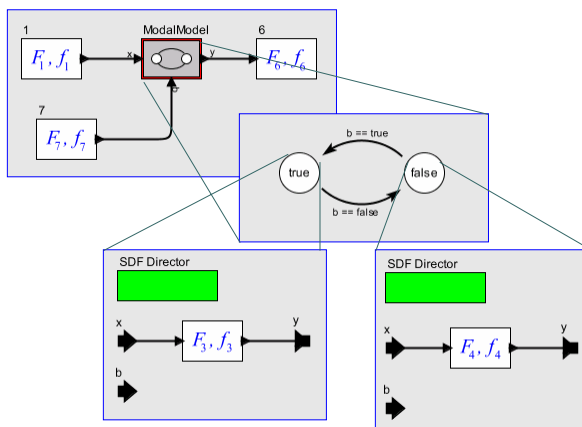
Heterochronous Dataflow (HDF) (Girault, Lee, and Lee, 1997)

Related to "parameterized dataflow" of Bhattacharya and Bhattacharyya (2001).

- An interconnection of actors.
- An actor is either SDF or HDF.
- If HDF, then the actor has:
 - a state machine
 - a refinement for each state
 - where the refinement is an SDF or HDF actor
- Operational semantics:
 - with the state of each state machine fixed, graph is SDF
 - in the initial state, execute one complete SDF iteration
 - evaluate guards and allow state transitions
 - in the new state, execute one complete SDF iteration
- HDF is decidable if state machines are finite
 - but complexity can be high

Lee 17: 11

If-Then-Else Using Heterochronous Dataflow



Imperative equivalent:

```

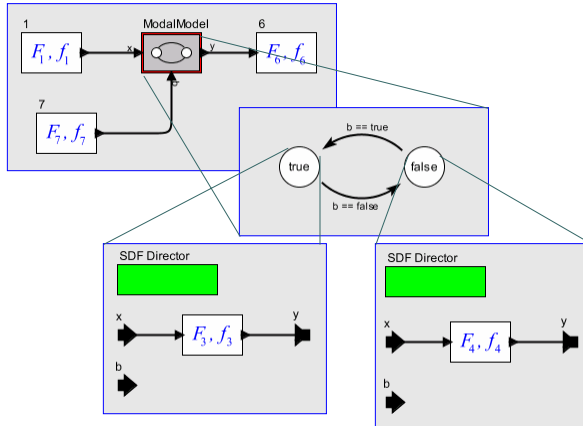
b = true;
while (true) {
    x = f1();
    if (b) {
        y = f3(x);
    } else {
        y = f4(x);
    }
    f6(y);
    b = f7();
}
    
```

Semantics of HDF:

- Execute SDF model for one complete iteration in current state
- Take state transitions to get a new SDF model.

Lee 17: 12

If-Then-Else Using Heterochronous Dataflow



Imperative equivalent:

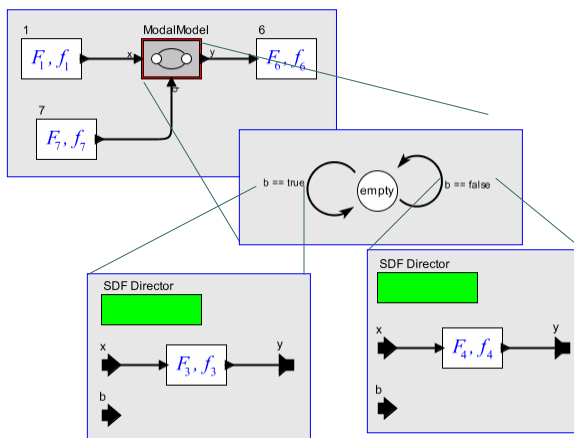
```

b = true;
while (true) {
    x = f1();
    if (b) {
        y = f3(x);
    } else {
        y = f4(x);
    }
    f6(y);
    b = f7();
}
    
```

Note that if these two refinements have the same production/consumption parameters, then this is simply hierarchical SDF, where one static schedule suffices.

Lee 17: 13

Hierarchical SDF Using Transition Refinements



Imperative equivalent:

```

while (true) {
    x = f1();
    b = f7();
    if (b) {
        y = f3(x);
    } else {
        y = f4(x);
    }
    f6(y);
}
    
```

This only works under rather narrow constraints:

- Exactly one outgoing transition from any state is enabled.
- The transition refinements on all transitions have the same production/consumption patterns.
- The state has no refinement.

Lee 17: 14

Conclusions and Open Issues

- Generalizations to SSDF improve expressiveness while preserving decidability.
- Usable languages for many of these extensions have yet to be created.