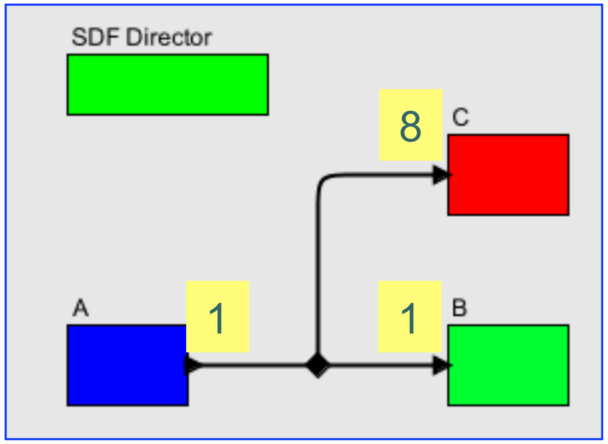
CS 5244: Introduction to Cyber Physical Systems

Unit 19: Dataflow Models 2 (Ch. 6)

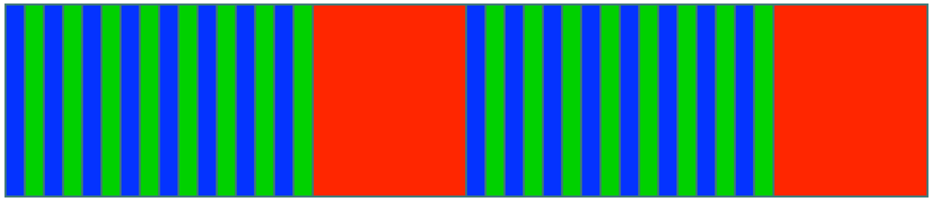
Instructor: Cheng-Hsin Hsu

Acknowledgement: The instructor thanks Profs. Edward A. Lee & Sanjit A. Seshia at UC Berkeley for sharing their course materials

Abstracted Version of the Spectrum Example

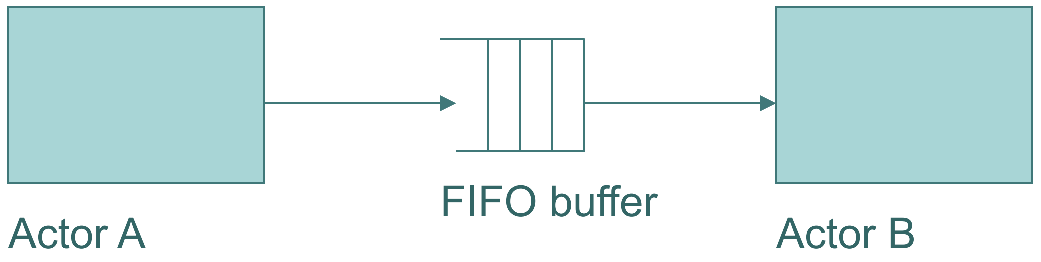


Suppose that C requires 8 data values from A to execute. Suppose further that C takes much longer to execute than A or B. Then a schedule might look like this:



…

Dataflow Models

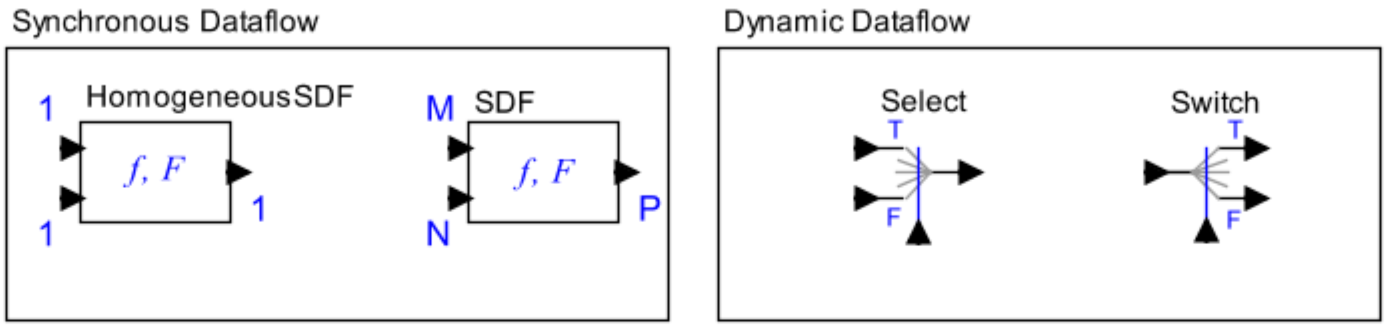


Buffered communication between concurrent components (*actors*).

An actor can fire whenever it has enough data (*tokens*) in its input buffers. It then produces some data on its output buffers.

In principle, buffers are unbounded. But for implementation on a computer, we want them bounded (and as small as possible).

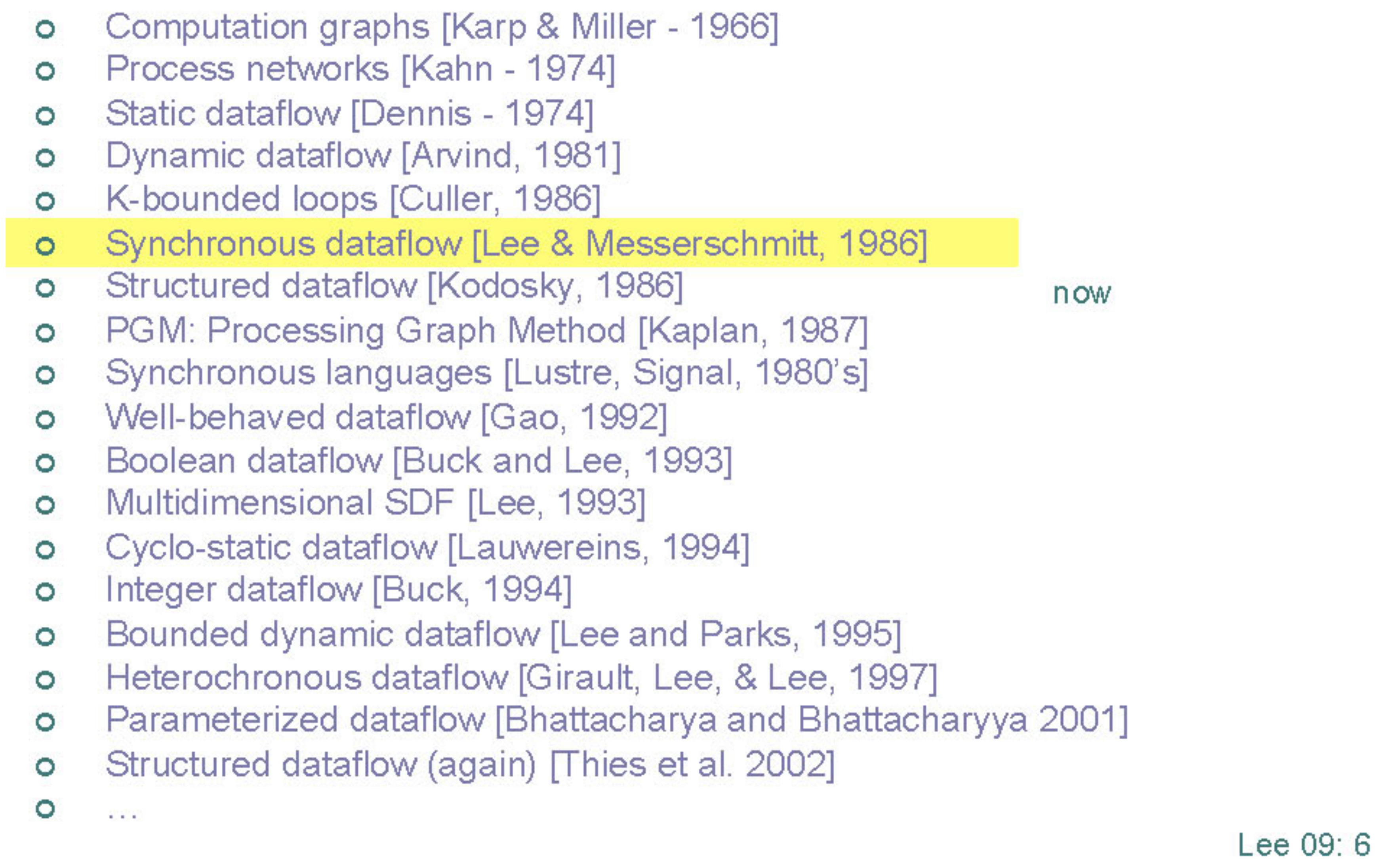
Dataflow



Each signal has form $x\colon \mathbb{N} \to R$. The function F maps such signals into such signals. The function f (the "firing function") maps prefixes of these signals into prefixes of the output. Operationally, the actor *consumes* some number of tokens and *produces* some number of tokens to construct the output signal(s) from the input signal(s). If the number of tokens consumed and produced is a constant over all firings, then the actor is called a *synchronous dataflow* (SDF) actor.

Firing rules: the number of tokens required to fire an actor.

Dataflow: many variants, still active area of research

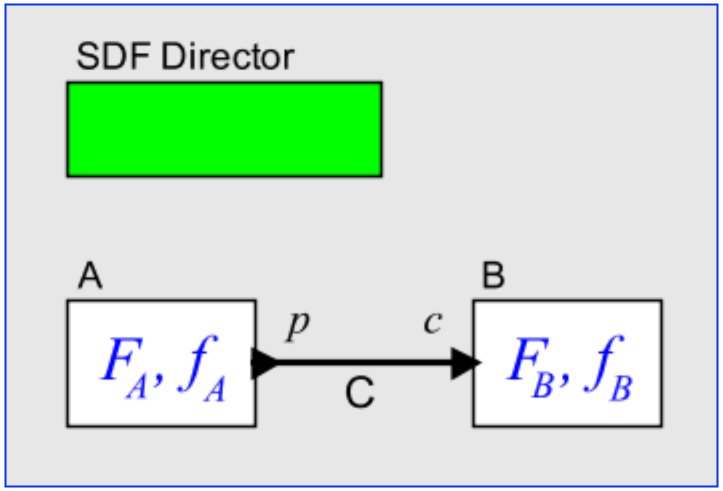


- Computation graphs [Karp & Miller - 1966]
- Process networks [Kahn - 1974]
- Static dataflow [Dennis - 1974]
- Dynamic dataflow [Arvind, 1981]
- K-bounded loops [Culler, 1986]
- Synchronous dataflow [Lee & Messerschmitt, 1986]
- Structured dataflow [Kodosky, 1986]
- PGM: Processing Graph Method [Kaplan, 1987]
- Synchronous languages [Lustre, Signal, 1980's]
- Well-behaved dataflow [Gao, 1992]
- Boolean dataflow [Buck and Lee, 1993]
- Multidimensional SDF [Lee, 1993]
- 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]
- Structured dataflow (again) [Thies et al. 2002]
- …

now

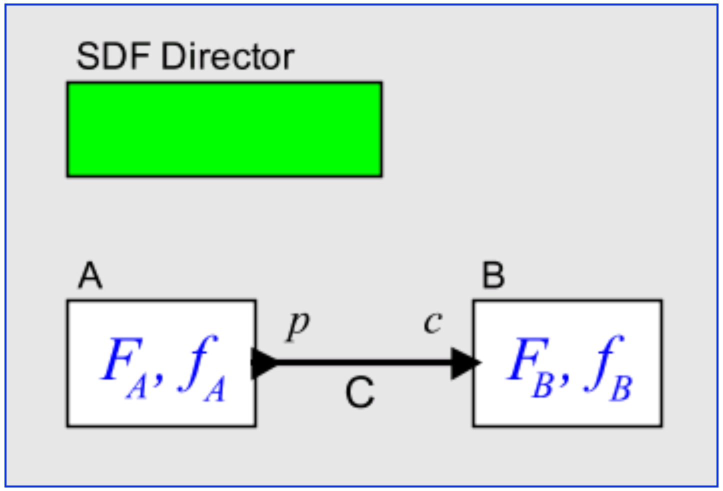
Lee 09: 6

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.

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 tokens produced and consumed on a connection C.

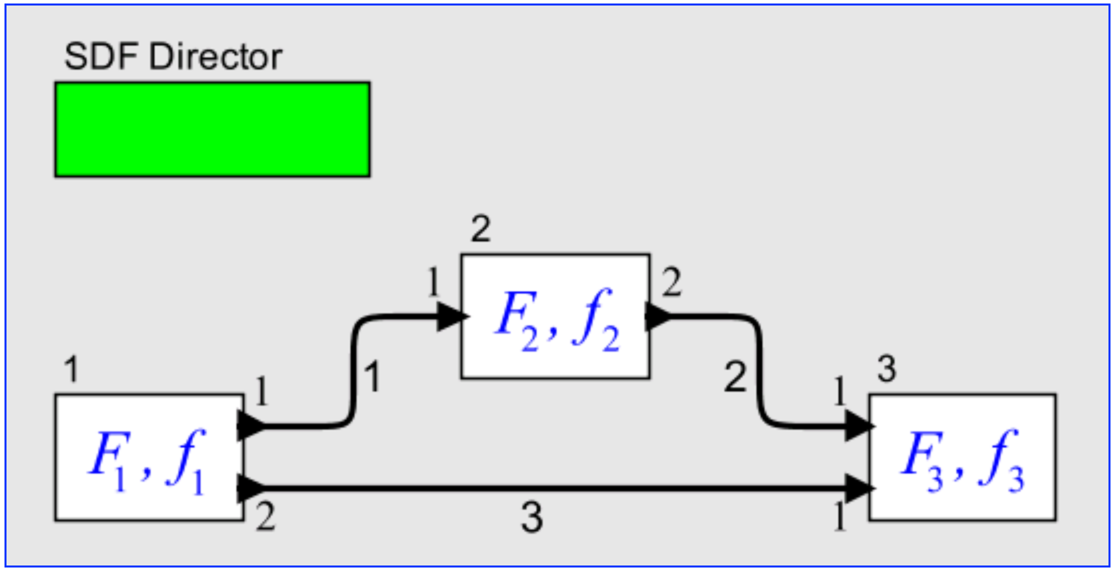
Then the system is *in balance* if for all connections C

$$q_A p_C = q_B c_C$$

where A produces tokens on C and B consumes them.

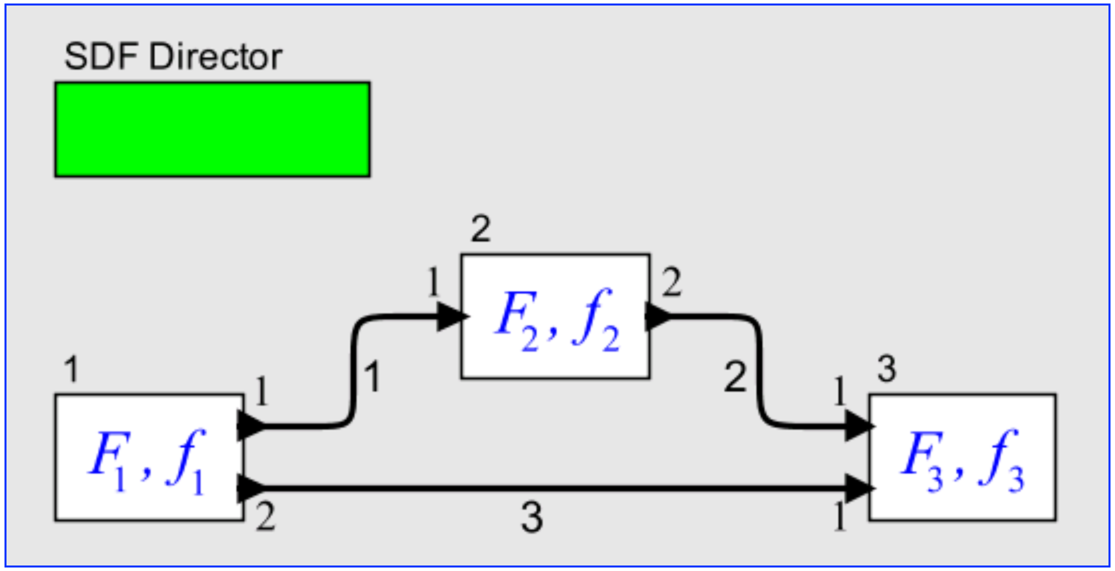
Example

Consider this example, where actors and arcs are numbered:



The balance equations imply that actor 3 must fire twice as often as the other two actors.

Compactly Representing the Balance Equations



production/consumption matrix

Connector 1

$$\Gamma = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 2 & -1 \\ 2 & 0 & -1 \end{bmatrix}$$

Actor 1

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

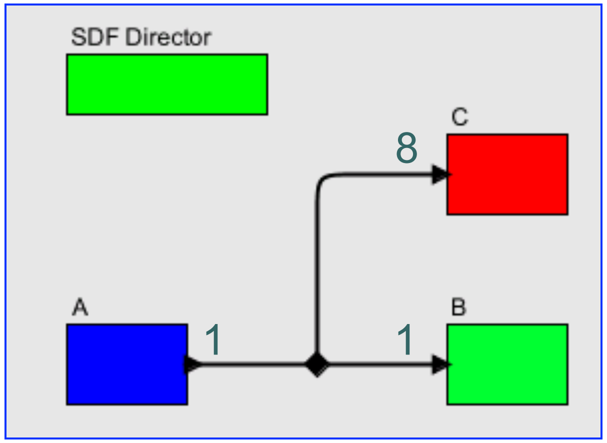
firing vector

balance equations

$$\Gamma q = \vec{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

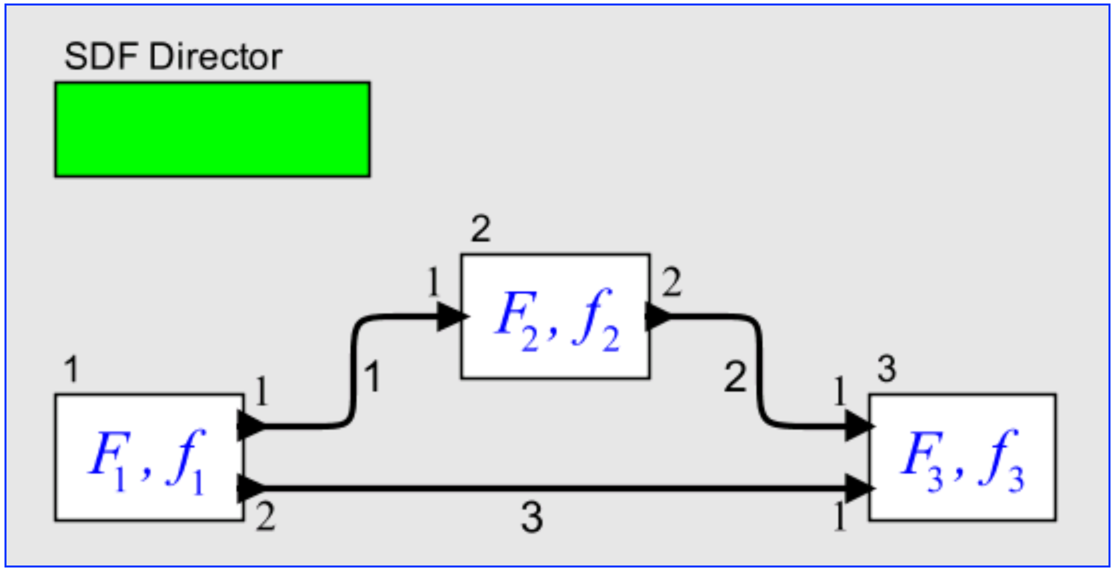
Question on material from the last time …

What is the production/consumption matrix in this case?



$$\Gamma = \begin{bmatrix} 1 & 0 & -1 \\ 1 & -8 & 0 \end{bmatrix}$$

Example

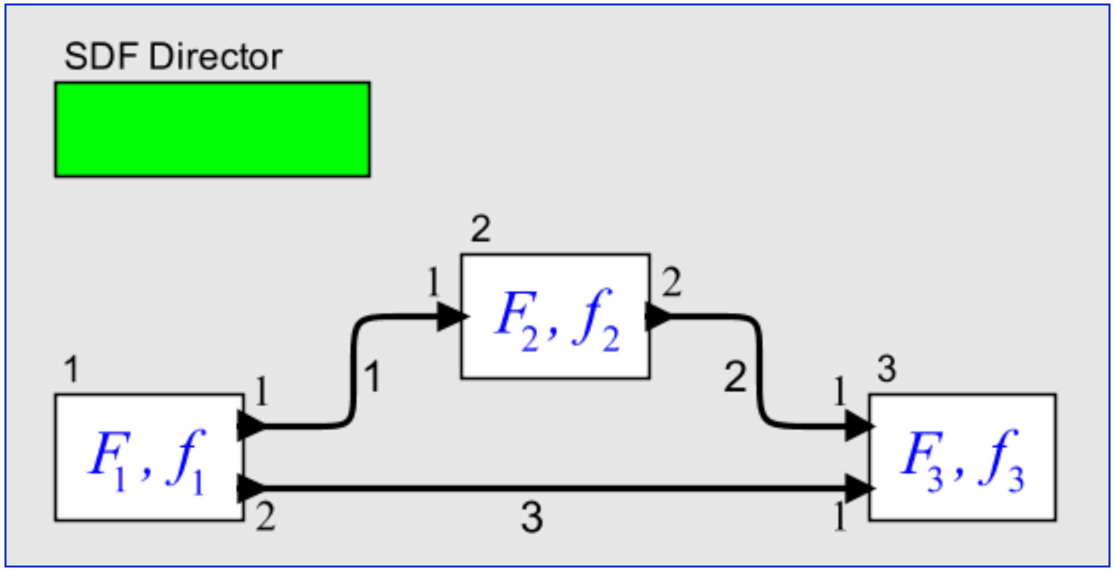


A solution to the balance equations:

$$q = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} \qquad \Gamma = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 2 & -1 \\ 2 & 0 & -1 \end{bmatrix} \qquad \Gamma q = \vec{0}$$

This tells us that actor 3 must fire twice as often as actors 1 and 2.

Example



But there are many solutions to the balance equations:

$$q = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} \quad q = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad q = \begin{bmatrix} 2 \\ 2 \\ 4 \end{bmatrix} \quad q = \begin{bmatrix} -1 \\ -1 \\ -2 \end{bmatrix} \quad q = \begin{bmatrix} \pi \\ \pi \\ 2\pi \end{bmatrix} \qquad \Gamma q = \vec{0}$$

For “well-behaved” models, there is a unique least positive integer solution.

Least Positive Integer Solution to the Balance Equations

Note that if p_C, c_C , the number of tokens produced and consumed on a connection C, are non-negative integers, then the balance equation,

$$q_A p_C = q_B c_C$$

implies:

- q_A is rational if and only if q_B is rational.
- q_A is positive if and only if q_B is positive.

Consequence: Within any connected component, if there is any non-zero solution to the balance equations, then there is a unique least positive integer solution.

Rank of a Matrix

The rank of a matrix Γ is the number of linearly independent rows or columns. The equation

$$\Gamma q = \vec{0}$$

is forming a linear combination of the columns of Γ. Such a linear combination can only yield the zero vector if the columns are linearly dependent (this is what is means to be linearly dependent).

If Γ has a columns and b rows, the rank cannot exceed $\min(a, b)$. If the columns or rows of Γ are re-ordered, the resulting matrix has the same rank as Γ.

Rank of the Production/Consumption Matrix

Let a be the number of actors in a connected graph. Then the *rank* of the production/consumption matrix Γ is $\leq a$. (why?)

If the model is a *spanning tree* (meaning that there are barely enough connections to make it connected) then Γ has a rows and $a-1$ columns.

Theorem [Lee-Messerschmitt' 87]: Its rank is $a-1$.

Exercise: Prove it. (Hint: use induction).

Corollary: the rank of any production/consumption matrix of a connected graph is either a or $a-1$. (why?)

Consistent Models

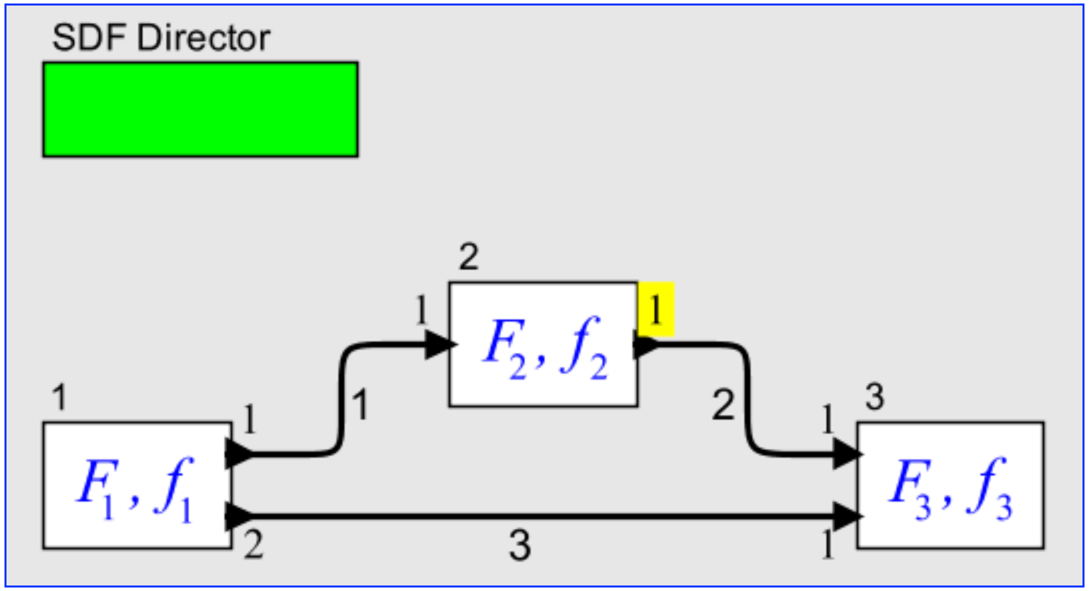
Let a be the number of actors in a connected model. The model is *consistent* if Γ has rank $a - 1$.

If the rank is a, then the balance equations have only a trivial solution (zero firings).

When Γ has rank $a - 1$, then the balance equations always have a non-trivial solution.

Example of an Inconsistent Model: No Non-Trivial Solution to the Balance Equations

$$\Gamma = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 2 & 0 & -1 \end{bmatrix}$$



This production/consumption matrix has rank 3, so there are no nontrivial solutions to the balance equations.

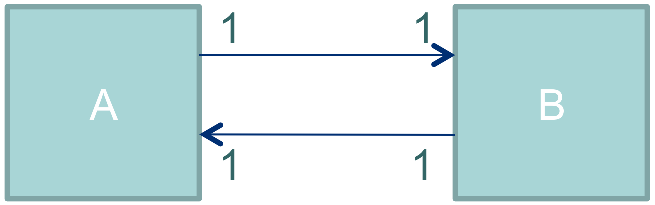
Note that this model can execute forever, but it requires unbounded memory.

Necessary and sufficient conditions

Consistency is a necessary condition to have a (bounded-memory) infinite execution.

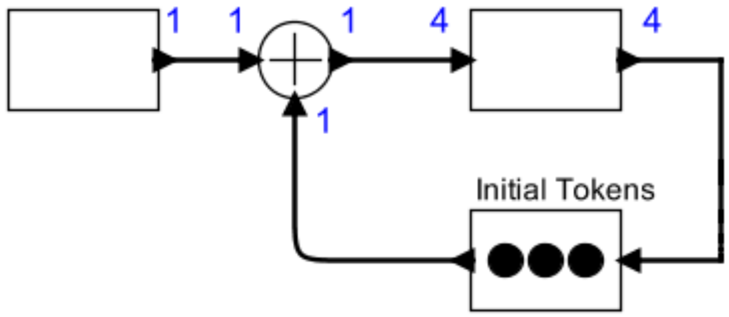
Is it sufficient?

Deadlock 1



Is this diagram consistent?

Deadlock 2



Some dataflow models cannot execute forever. In the above model, the feedback loop injects initial tokens, but not enough for the model to execute.

SDF: from static analysis to scheduling

Given: SDF diagram

Find: a bounded-buffer schedule, if it exists

Step 0: check whether diagram is consistent. If not, then no bounded-buffer schedule exists.

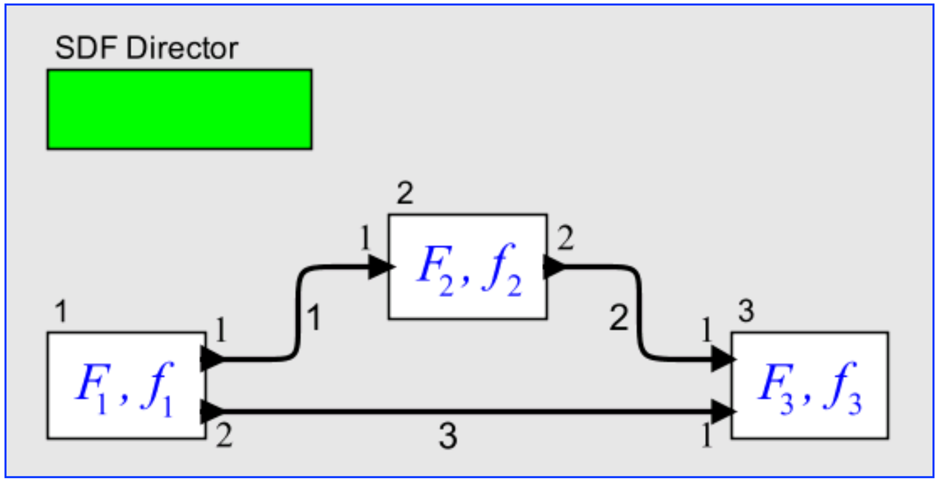
Step 1: find an integer solution to $\mathbf{\Gamma}\boldsymbol{q}=\boldsymbol{0}$.

Step 2: “decompose” the solution q into a schedule, making sure buffers never become negative.

Step 2: "decomposing" the firing vector

Example 1:

Schedule = (1;2;3;3)

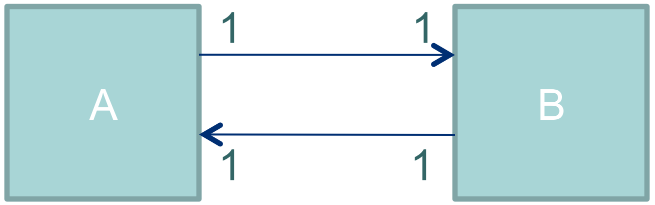


$$b=\begin{bmatrix}0\\0\\0\end{bmatrix} \xrightarrow{\text{Fire 1}} b=\begin{bmatrix}1\\0\\2\end{bmatrix} \xrightarrow{\text{Fire 2}} b=\begin{bmatrix}0\\2\\2\end{bmatrix} \xrightarrow{\text{Fire 3}} b=\begin{bmatrix}0\\1\\1\end{bmatrix} \xrightarrow{\text{Fire 3}} b=\begin{bmatrix}0\\0\\0\end{bmatrix}$$

$$q=\begin{bmatrix}1\\1\\2\end{bmatrix} \quad q=\begin{bmatrix}0\\1\\2\end{bmatrix} \quad q=\begin{bmatrix}0\\0\\2\end{bmatrix} \quad q=\begin{bmatrix}0\\0\\1\end{bmatrix} \quad q=\begin{bmatrix}0\\0\\0\end{bmatrix}$$

Step 2: "decomposing" the firing vector

Example 2:



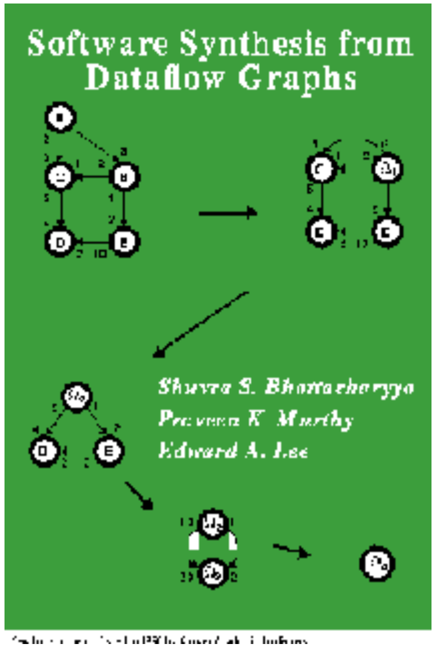
What happens if we try to run the previous procedure on this example?

So, we have both necessary and sufficient conditions for scheduling SDF graphs.

A Key Question: If More Than One Actor is Fireable in Step 2, How do I Select One?

Optimization criteria that might be applied:

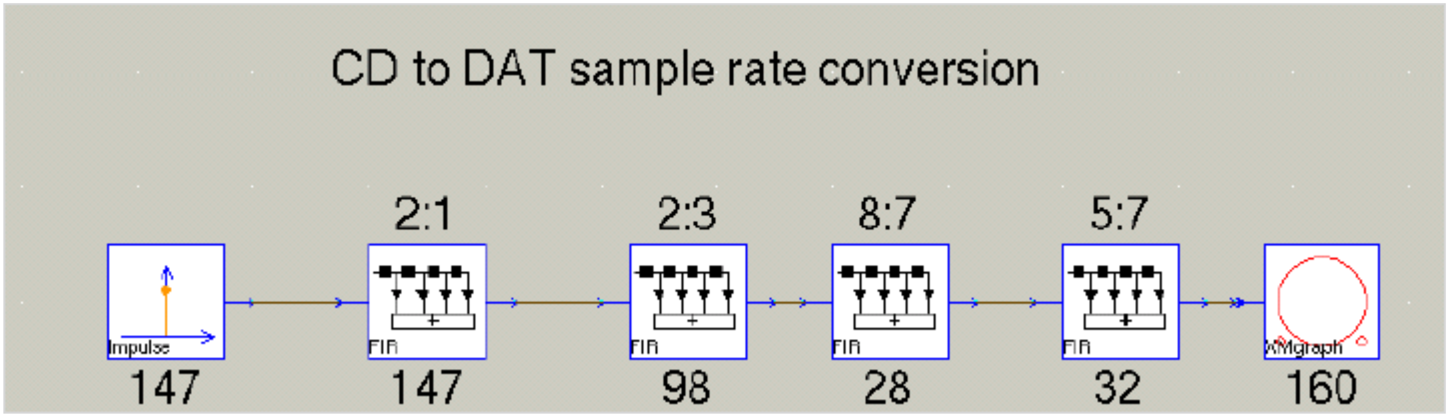
- Minimize buffer sizes.
- Minimize the number of actor activations.
- Minimize the size of the representation of the schedule (code size).



See S. S. Bhattacharyya, P. K. Murthy, and E. A. Lee, *Software Synthesis from Dataflow Graphs,* Kluwer Academic Press, 1996.

Beyond our scope here, but hints that it's an interesting problem…

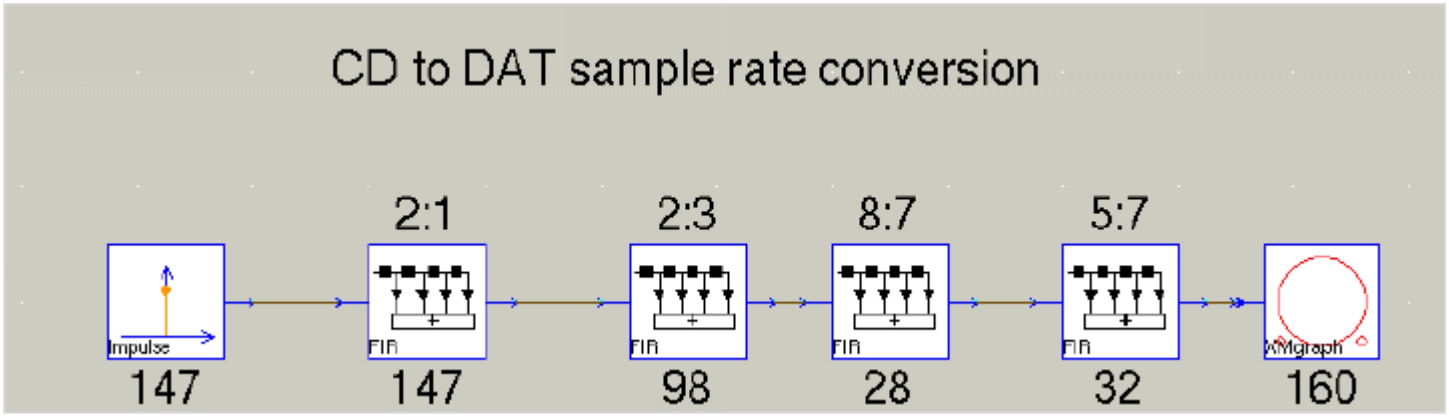
Minimum Buffer Schedule



```
A B A B C A B C A B A B C A B C D E A F F F F F B A B C A B C A B A B C D E
A F F F F F B C A B A B C A B C A B A B C D E A F F F F F B C A B A B C A B C
D E A F F F F F B A B C A B C A B A B C A B C D E A F F F F F B A B C A B C A
B A B C D E A F F F F F B C A B A B C A B C A B A B C D E A F F F F F E B C A
F F F F F B A B C A B C D E A F F F F F B A B C A B C A B A B C A B C D E A F
F F F F B A B C A B C A B A B C D E A F F F F F B C A B A B C A B C A B A B C
D E A F F F F F B C A B A B C A B C D E A F F F F F B A B C A B C A B A B C A
B C D E A F F F F F B A B C A B C A B A B C D E A F F F F F E B C A F F F F F B
A B C A B C A B A B C D E A F F F F F B C A B A B C A B C D E A F F F F F B A
B C A B C A B A B C A B C D E A F F F F F B A B C A B C A B A B C D E A F F F
F F B C A B A B C A B C A B A B C D E A F F F F F B C A B A B C A B C D E A F
F F F F B A B C A B C A B A B C A B C D E A F F F F F E B A F F F F F B C A B C
A B A B C D E A F F F F F B C A B A B C A B C A B A B C D E A F F F F F B C A
B A B C A B C D E A F F F F F B A B C A B C A B A B C A B C D E A F F F F F B
A B C A B C A B A B C D E A F F F F F B C A B A B C A B C A B A B C D E A F
F F F F B C A B A B C A B C D E F F F F F E F F F F F
```

Source: Shuvra Bhattacharyya

Scheduling Tradeoffs
(Bhattacharyya, Parks, Pino)

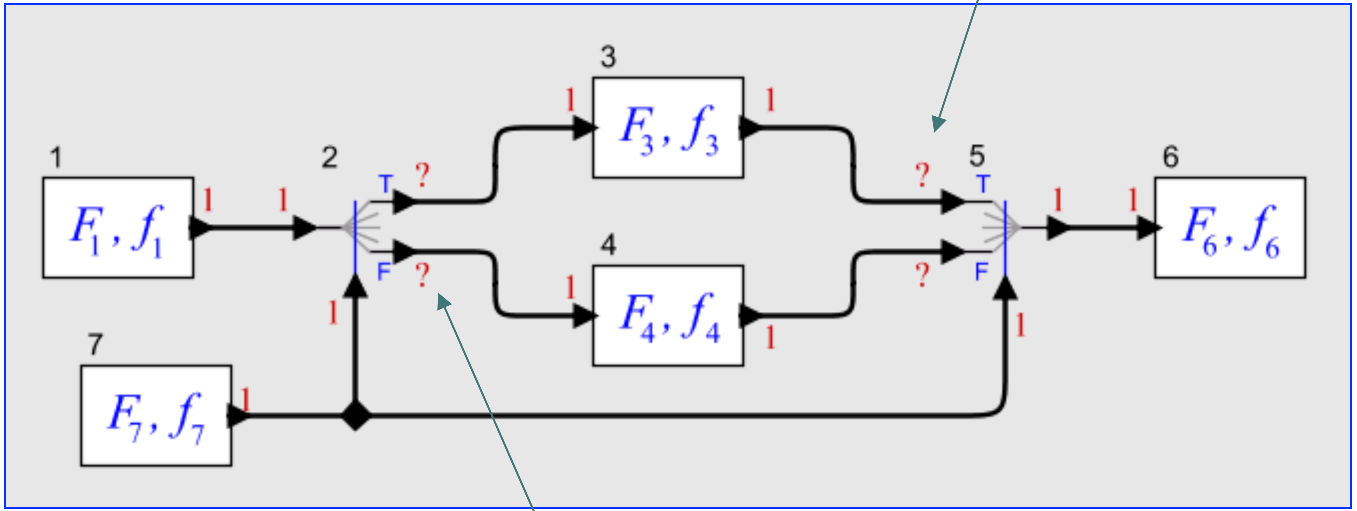


Scheduling strategy	Code	Data
Minimum buffer schedule, no looping	13735	32
Minimum buffer schedule, with looping	9400	32
Worst minimum code size schedule	170	1021
Best minimum code size schedule	170	264

Source: Shuvra Bhattacharyya

Dynamic Dataflow

What consumption rate?



What production rate?

Imperative equivalent:

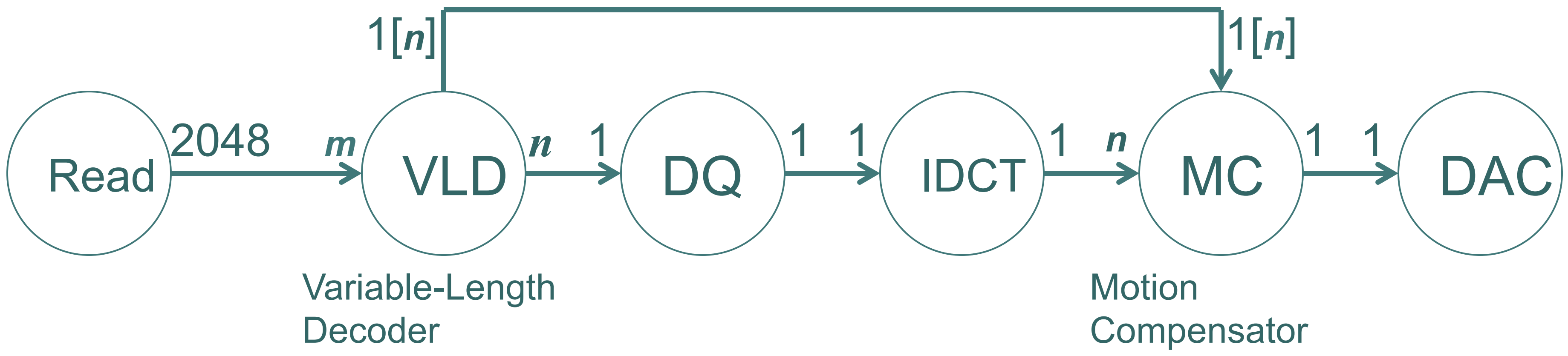
```
while (true) {
    x = f1();
    b = f7();
    if (b) {
      y = f3(x);
    } else {
      y = f4(x);
    }
    f6(y);
}
```

The if-then-else model is not SDF. But we can clearly give a bounded *quasi-static* schedule for it:

(1, 7, 2, b?3, !b?4, 5, 6)

guard

Another example: H.263 video codec



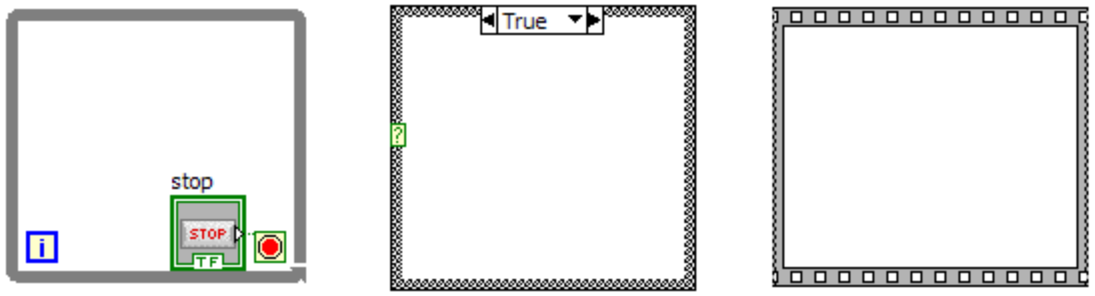
[Wiggers et al., RTAS' 08] conduct buffer analysis for such dynamic dataflow models.

Facts about (general) dynamic dataflow

- Whether there exists a schedule that does not deadlock is undecidable.
- Whether there exists a schedule that executes forever with bounded memory is undecidable.

Undecidable means that there is no algorithm that can answer the question in finite time for all finite models.

Structured Dataflow



LabVIEW uses homogeneous SDF augmented with syntactically constrained forms of feedback and rate changes:

- While loops
- Conditionals
- Sequences

LabVIEW models are decidable.

Many other concurrent MoCs have been explored

- (Kahn) process networks
- Communicating sequential processes (rendezvous)
- Time-driven models
- More dataflow variants:
  - cyclostatic
  - Heterochronous
  - …
- Petri nets
- …

Reading

E.A. Lee and D.G. Messerschmitt. *Synchronous data flow*. In *Proceedings of the IEEE*. 1987.

S.S. Battacharyya, P.K. Murthy, and E.A. Lee. *Software Synthesis from Dataflow Graphs*. Kluwer, 1996.

M. Wiggers, M. Bekooij, and G. Smit. *Buffer Capacity Computation for Throughput Constrained Streaming Applications with Data-Dependent Inter-Task Communication*. In RTAS' 08, IEEE, 2008.