Data flow models

Peter Marwedel
TU Dortmund,
Informatik 12
## Models of computation considered in this course

<table>
<thead>
<tr>
<th>Communication/local computations</th>
<th>Shared memory</th>
<th>Message passing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undefined components</td>
<td>Plain text, use cases</td>
<td>Synchronous</td>
</tr>
<tr>
<td>Communicating finite state machines</td>
<td>StateCharts</td>
<td>SDL</td>
</tr>
<tr>
<td>Data flow</td>
<td>Scoreboarding + Tomasulo Algor. (Comp.Archict.)</td>
<td>Kahn networks, SDF</td>
</tr>
<tr>
<td>Petri nets</td>
<td></td>
<td>C/E nets, P/T nets, …</td>
</tr>
<tr>
<td>Discrete event (DE) model</td>
<td>VHDL, Verilog, SystemC, …</td>
<td>Only experimental systems, e.g. distributed DE in Ptolemy</td>
</tr>
<tr>
<td>Von Neumann model</td>
<td>C, C++, Java</td>
<td>C, C++, Java with libraries CSP, ADA</td>
</tr>
</tbody>
</table>
Data flow as a “natural” model of applications

Example: Video on demand system

- Customer List
- Admission control
- Scheduler
- File System
- Storage Control
- Network Address
- Viewers
- Storage Subsystem
- Network Interface
- Video Data
- Video on Demand System

www.ece.ubc.ca/~irenek/techpaps/vod/vod.html
Definition: Data flow modeling is … “the process of identifying, modeling and documenting how data moves around an information system. Data flow modeling examines

- processes (activities that transform data from one form to another),
- data stores (the holding areas for data),
- external entities (what sends data into a system or receives data from a system, and
- data flows (routes by which data can flow”).

Kahn process networks (KPN)

- Each component is modeled as a program/task/process, (underlying FSM is inconvenient: possibly many states)

- Communication is by FIFOs; no overflow considered
  - writes never have to wait,
  - reads wait if FIFO is empty.

- Only one sender and one receiver per FIFO
  - no SDL-like conflicts at FIFOs
Example

Process `f(in int u, in int v, out int w) {`  
`int i; bool b = true;`  
`for (;;) {`  
  `i = b ? read(u) : read(v);`  
  `//read returns next token in FIFO, waits if empty`  
  `send (i,w);`  
  `//writes a token into a FIFO w/o blocking`  
  `b = !b;`  
`}`
Properties of Kahn process networks

- Communication is only via channels (no shared variables)
- Mapping from $\geq 1$ input channel to $\geq 1$ output channel possible;
- Channels transmit information within an unpredictable but finite amount of time;
- In general, execution times are unknown.
Key beauty of KPNs (1)

- A process cannot check for available data before attempting a read (wait).

\[
\text{if nonempty(p1) then read(p1) else if nonempty(p2) then read(p2);}
\]

- A process cannot wait for data for >1 port at a time.

\[
\text{read(p1|p2);}
\]

Processes have to commit to wait for data from a particular port;
Key beauty of KPNs (2)

Therefore, the order of reads does not depend on the arrival time (but may depend on data).

Therefore, Kahn process networks are **determinate** (!); for a given input, the result will always the same, regardless of the speed of the nodes.

Many applications in embedded system design: Any combination of fast and slow simulation & hardware prototypes always gives the same result.
Computational power and analyzability

- It is a challenge to schedule KPNs without accumulating tokens.
- KPNs are Turing-complete (anything which can be computed can be computed by a KPN).
- KPNs are computationally powerful, but difficult to analyze (e.g. what’s the maximum buffer size?)
- Number of processes is static (cannot change).
More information about KPNs

- See also S. Edwards: http://www.cs.columbia.edu/~sedwards/classes/2001/w4995-02/presentations/dataflow.ppt
Less computationally powerful, but easier to analyze:

**Synchronous data flow (SDF).**

- **Synchronous**
  = global clock controlling “firing” of nodes

- Again using asynchronous message passing
  = tasks do not have to wait until output is accepted.
(Homogeneous-)
Synchronous data flow (SDF)

- Nodes are called **actors**.
- Actors are **ready**, if the necessary number of input tokens exists and if enough buffer space at the output exists.
- Ready actors **can fire** (be executed).

- Execution takes a **fixed, known time**.
Actually, this is a case of **homogeneous** synchronous data flow models (HSDF): # of tokens per firing the same.
(Non-homogeneous-) Synchronous data flow (SDF) (1)

In the general case, a number of tokens can be produced/consumed per firing

A ready, can fire (does not have to)
(Non-homogeneous-)
Synchronous data flow (SDF) (2)

In the general case, a number of tokens can be produced/consumed per firing

B ready, can fire
(Non-homogeneous-)
Synchronous data flow (SDF) (3)

In the general case, a number of tokens can be produced/ consumed per firing

A ready, can fire
(Non-homogeneous-) Synchronous data flow (SDF) (4)

In the general case, a number of tokens can be produced/consumed per firing.

B ready, can fire
(Non-homogeneous-) Synchronous data flow (SDF) (5)

In the general case, a number of tokens can be produced/consumed per firing.

B ready, can fire
(Non-homogeneous-)
Synchronous data flow (SDF) (6)

In the general case, a number of tokens can be produced/ consumed per firing

1 period complete, A ready, can fire
Actual modeling of buffer capacity

The capacity of buffers can be modeled easier: as a \textit{backward} edge where (initial number of tokens = buffer capacity).

Firing rate depends on # of tokens …
Multi-rate models & balance equations (one for each channel)

\[ f_A N = f_B M \]

Decidable:
- buffer memory requirements
- deadlock

Scheduler statically

Adopted from: ptolemy.eecs.berkeley.edu/presentations/03/streamingEAL.ppt
Parallel Scheduling of SDF Models

SDF is suitable for automated mapping onto parallel processors and synthesis of parallel circuits.

Many scheduling optimization problems can be formulated.

Source: ptolemy.eecs.berkeley.edu/presentations/03/streamingEAL.ppt
Expressiveness of data flow MoCs

CSDF = Cyclo static data flow (rates vary in a cyclic way)

[S. Stuijk, 2007]
The expressiveness/analyzability conflict

[S. Stuijk, 2007]