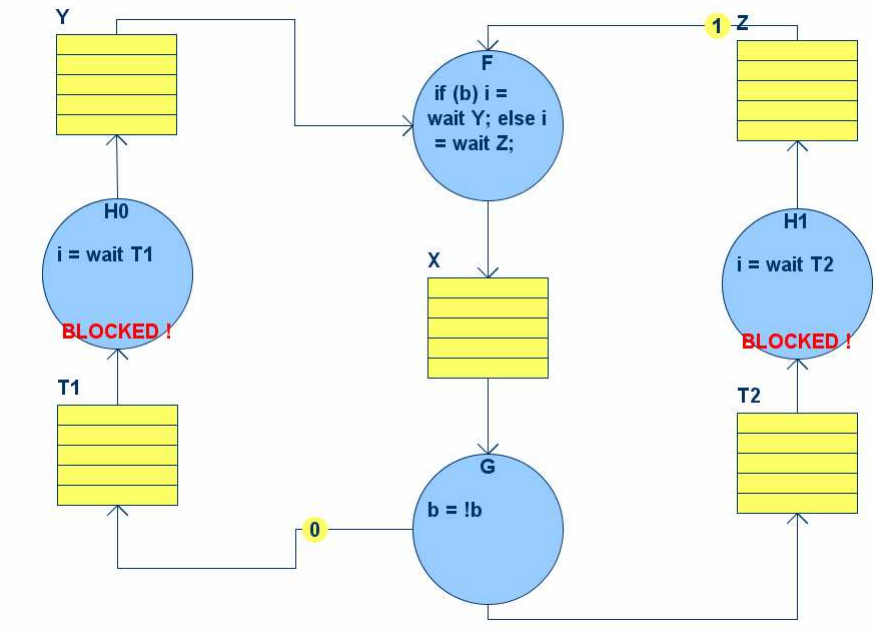
Embedded Systems

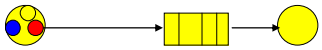
Bernd Finkbeiner
Calogero Zarba

Sommersemester 2007

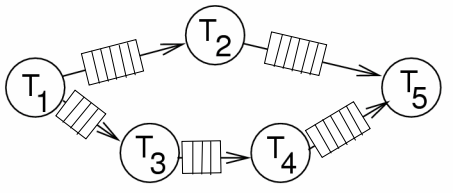


Asynchronous message passing:
Kahn process networks

For asynchronous message passing: communication between tasks is buffered



Special case: Kahn process networks: executable task graphs; Communication is assumed to be via infinitely large FIFOs



Properties of Kahn process networks

- Each node corresponds to one program/task;
- communication is only via channels;
- channels include FIFOs as large as needed;
- one producer and one consumer;
- channels transmit information within an unpredictable but finite amount of time;
- send operations are non-blocking, reads are blocking;
- each node may be either waiting for input on **one** of its input channels or executing;
- semantics:
  - associated with each node is a continuous function from input sequences to output sequences;
  - this defines a unique assignment of sequences to channels.

Example

☞ Model of parallel computation used in practice (e.g. at Philips).

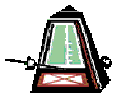
```
Process f (in int u, in int v, out int w) {
   int i; bool b = true;
   for (;;) {
         i = b ? wait (u) : wait (v); // wait returns next token in FIFO, blocks if empty
         printf("%i\n", i);
         send (i, w); // writes a token into a FIFO w/o blocking
         b = !b;
   }
}
```

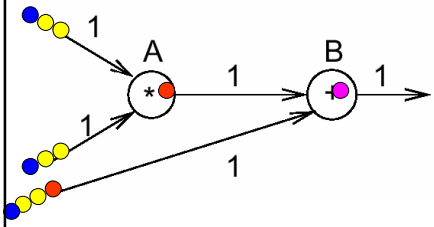
- A process can not check whether data is available before attempting a read
- A process can not wait for data on more than one port at a time
- Therefore, order of reads, writes depend only on data, not its arrival time
- It is a challenge to schedule KN without accumulating tokens.

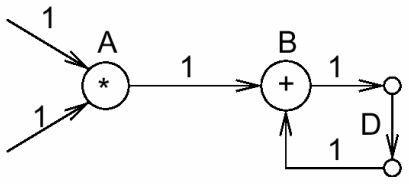
☞easier for SDF

Asynchronous message passing: Synchronous data flow (SDF)

- Asynchronous message passing= tasks do not have to wait until output is accepted.
- Synchronous data flow = all tokens are consumed at the same time.







SDF model allows static scheduling of token production and consumption.

SDF Scheduling

Two steps:

1. Establish relative execution rates
   (by solving linear equations)

2. Determine periodic schedule
   (by simulating a single round)

Example Scheduling Algorithm (Lee/Messerschmitt)

1. Compute rate q_α for each node α
2. Form an arbitrarily ordered list L of all nodes in the system
3. For each node $\alpha \in L$, schedule α if it has been scheduled less than q_α times and can run, trying each node once
4. If each node α has been scheduled q_α times, STOP
5. If no node can be scheduled, report DEADLOCK
6. Else, go to step 3.

Coming Up May 22: Midterm Exam