Synchronous Dataflow

• Well-suited to multirate signal processing

\[
\begin{bmatrix}
1 & -1 & 0 & 0 & 0 & 0 \\
0 & 2 & -3 & 0 & 0 & 0 \\
0 & 0 & 4 & -7 & 0 & 0 \\
0 & 0 & 0 & 5 & -7 & 0 \\
0 & 0 & 0 & 0 & 4 & -1 \\
\end{bmatrix}
\times
\begin{bmatrix}
147 \\
147 \\
98 \\
56 \\
40 \\
160 \\
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
\end{bmatrix}
\]

Static Scheduling

Synchronous communication with a fixed number of tokens transferred

♦ Balance equations determine repetitions

♦ Periodic schedule is computed statically
**Dynamic Scheduling**

Asynchronous communication between independent tasks

- Repetition rates are not known exactly
- Dynamic, run-time scheduling is required
Rate-Monotonic Scheduling

- Periodic tasks are independent
  \[ T_i \text{ Period} \]
  \[ C_i \text{ Execution time} \]
- Rate-monotonic priority assignment is optimal:
  \[ T_1 \leq T_2 \leq \cdots \leq T_N \]
  ◦ Only periods \( T_i \) are needed for scheduling
  ◦ Execution times \( C_i \) are needed only to prove feasibility
  ◦ Best-effort scheduling with imprecise execution times

Preemptive Real-Time Scheduling

- Fixed priorities
  - Limited processor utilization 70%
  + Simple, efficient implementation
  + Graceful degradation in overload situations
- Dynamic priorities
  + Better processor utilization 100%
  - Complex implementation
  - Unpredictable behavior in overload situations
### Priority Inheritance Protocols

- Periodic tasks contend for exclusive access to shared resources
- $B_i$ Blocking time
- Processor utilization bound

\[
\forall i \in [1\cdots N] \min_{(k,l) \in R_i} \sum_{j=1}^{i-1} \frac{C_j}{lT_k} \left\lfloor \frac{lT_k}{T_j} \right\rfloor + \frac{C_i}{lT_k} + \frac{B_i}{lT_k} \leq 1
\]
Non-Preemptive Scheduling

- CPU is a shared resource
- Blocking time is the execution time of one sub-task
  \[ B_i = \max_{(j,k) \in S_i} C_{j,k} \]
  \[ S_i = \{(j,k) | i + 1 \leq j \leq N, 1 \leq k \leq N_j \} \]
- Accounting for context switch costs
  \[ B_i = 2\Delta + \max_{(j,k) \in S_i} C_{j,k} \]

Multithreaded Architectures

- Massively parallel, general-purpose architectures
  - High communication latency
  - Synchronization
- Hybrid Dataflow/von Neumann machines
  - Tolerate latency
  - Exploit pipelines and caches
  Examples: TAM, *T
Real-Time Extensions to TAM

- Static, periodic schedule replaces the dynamic continuation vector
- Tasks may yield control after any sub-task instead of running until the continuation vector is empty
- Tasks are prioritized

Real-Time Multithreaded Execution

![Dataflow graph and static schedule diagram]

- Current thread
- Enabled threads (sorted by priority)
- Suspended threads
- Preemption vector
- Dataflow graph

11

12
Hybrid Static/Dynamic Scheduling

• Static, periodic schedules for each independent dataflow graph

• Dynamic, non-preemptive rate-monotonic scheduling for system of periodic tasks

Conclusions

• Sufficient conditions for feasible non-preemptive scheduling

• Efficient real-time execution model
Future Work

• Exact characterization of non-preemptive rate-monotonic scheduling

• Explore other system representations
  • Cyclo-static dataflow to reduce blocking times
  • Clustering to reduce run-time scheduling overhead