EECE 276
Embedded Systems

Techniques:
Petri nets
Petri nets

- Graphical representation for concurrent systems
- Ingredients:
  - “Places”: process states, data stores, etc.
  - “Transitions”: changes, operations, etc.
  - “Marking”: initial “data count” for places
- Firing: a transition fires if it has sufficient input marks. Marks from the input places are removed and moved to the output places of the transition.
Petri nets

Transition table

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>After</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Petri net example

A more complex sequence and its firing table

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_0$</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$m_1$</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>$m_2$</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$m_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>$m_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>
Petri nets represent control flow

Sequencing
Branching
Looping
Petri nets represent concurrency

Two processes + one semaphore
Petri nets and deadlocks

Typical deadlock situation
Petri Net Example

Process A

- $P_1$: (ready to send)
- $P_2$: (buffer in use)
- $P_3$: (buffer in use)
- $P_4$: (waiting for ack.)
- $P_5$: (received)
- $P_6$: (ack. received)
- $P_7$: (receive ack.)
- $P_8$: (ack. sent)

- $t_1$: (output)
- $t_2$: (input)
- $t_3$: (send ack.)
- $t_4$: (receive ack.)
- $t_5$: (produce)

Process B

- $P_3$: (ready to receive)
- $P_5$: (consume)
Shortcomings of Petri nets

- Very simple building blocks lead to complex nets
- Extensions:
  - Enabling/firing separated
  - Tokens not only for control but also as data
  - Attributes for transitions
    - Mean/variance of execution time
    - Transition probability