Announcements

• Checkpoint 1 is due next Tuesday
• My office hours next week will be on Thursday or by appointment.
• Slight reordering of lectures – allocation today, datapath components next Tuesday
Question of the day

• What would you rather have:
  – a perfect allocator and a switch with no speedup, or
  – an average allocator and a switch with a speedup of 2?
Basic Router Architecture

- Router
- VC Allocator
- Switch Allocator
- Input Unit
- Output Unit
- Switch
- IC
Arbiters & Allocators

• Resolve contention for single resource or a set of resources

• Resources
  – Switch ports
  – Virtual Channels
  – Physical Channels
  – Access to a shared routing table

• request → grant
  – Sometimes with “hold”
Definitions

- **Arbiter**
  - Allocates a single resource to one requester of a set of active requesters.

- **Allocator**
  - Allocates a set of resources (e.g., an input port and an output port of a switch) to each of several requesters.
  - Goal is to satisfy as many requesters as possible
    - Maximal matching – can’t satisfy any more requesters without taking away a resource already granted
    - Maximum matching – largest possible number of requesters satisfied
Fairness

- Local vs Global
- Parking lot example
- Different type of fairness
  - Weak
  - Strong
  - FIFO
Recall the Arbiter from our simple router example.
Fixed Priority
Priority

Diagram showing the priority logic circuit with inputs $p_i$, $r_i$, and $c_{i+1}$, and outputs $g_i$. The circuit includes AND and OR gates.
Setting Priority

- Randomly
- Rotating
- One past last granted

- What type of fairness do each of these have?
Matrix Arbiter

EE 382C - S11 - Lecture 11
Queuing Arbiter

![Queuing Arbiter Diagram]

- Timer
- Stamp
- Index
- Timestamp
- Comparator Tree
- Decode

Variables:
- $r_0, g_0, r_{n-1}, g_{n-1}$
- $0, t_0, (n-1), t_{n-1}$
- $i_w$
1. $g_{ij} \rightarrow r_{ij}$
2. One grant per resource
3. One grant per requestor
Matrix View

\[
\begin{array}{ccc}
  r_{00} & r_{01} & r_{02} \\
  r_{10} & r_{11} & r_{12} \\
  r_{20} & r_{21} & r_{22} \\
  r_{30} & r_{31} & r_{32} \\
\end{array}
\quad
\begin{array}{ccc}
  g_{00} & g_{01} & g_{02} \\
  g_{10} & g_{11} & g_{12} \\
  g_{20} & g_{21} & g_{22} \\
  g_{30} & g_{31} & g_{32} \\
\end{array}
\]

\[
\begin{array}{cccc}
  1 & 1 & 0 & 1 & 0 & 0 \\
  1 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 1 & 0 & 0 & 1 \\
  1 & 0 & 1 & 0 & 0 & 0 \\
\end{array}
\]

Graph View

(a)

(b)
Maximum Matching
Augmenting Path Algorithm
Maximum Matching
Augmenting Path Algorithm
Input-First Separable Allocator

\begin{align*}
\text{Arb} & \quad x_{00} \\
\text{Arb} & \quad x_{01} \\
\text{Arb} & \quad x_{02} \\
\text{Arb} & \quad x_{10} \\
\text{Arb} & \quad x_{11} \\
\text{Arb} & \quad x_{12} \\
\text{Arb} & \quad x_{20} \\
\text{Arb} & \quad x_{21} \\
\text{Arb} & \quad x_{22} \\
\text{Arb} & \quad x_{30} \\
\text{Arb} & \quad x_{31} \\
\text{Arb} & \quad x_{32} \\
\end{align*}

\begin{align*}
g_{00} & \\
g_{10} & \\
g_{20} & \\
g_{30} & \\
g_{01} & \\
g_{11} & \\
g_{21} & \\
g_{31} & \\
g_{02} & \\
g_{12} & \\
g_{22} & \\
g_{32} & \\
\end{align*}
Input First

- Pick a 1 in each row
- Then pick a 1 in each column

```
1 1 0
1 0 0
0 0 1
1 0 1

1 0 0
1 0 0
0 0 1
1 0 0
```
Output-First Separable Allocator

```
\begin{align*}
\text{Arb} & \quad y_{00} \quad g_{00} \\
r_{00} \\
r_{10} \\
r_{20} \\
r_{30} \\
\text{Arb} & \quad y_{10} \quad g_{01} \\
\text{Arb} & \quad y_{20} \quad g_{02} \\
\text{Arb} & \quad y_{30} \\
\text{Arb} & \quad y_{01} \quad g_{10} \\
\text{Arb} & \quad y_{11} \quad g_{11} \\
\text{Arb} & \quad y_{21} \quad g_{12} \\
\text{Arb} & \quad y_{31} \\
\text{Arb} & \quad y_{02} \quad g_{20} \\
\text{Arb} & \quad y_{12} \quad g_{21} \\
\text{Arb} & \quad y_{22} \quad g_{22} \\
\text{Arb} & \quad y_{32} \\
\text{Arb} & \quad g_{30} \\
\text{Arb} & \quad g_{31} \\
\text{Arb} & \quad g_{32}
\end{align*}
```
Output First

- Pick a 1 in each column
- Then pick a 1 in each row

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

1 0 1 0 0 0 0 0 0 0 0 0
Iterative Allocation

- After each round
  - remove conflicting requests
  - Retry remaining Grants

\[
\begin{array}{cccc}
1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 \\
\end{array}
\hspace{1cm}
\begin{array}{cccc}
1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 \\
\end{array}
\hspace{1cm}
\begin{array}{cccc}
0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
\end{array}
\]
Iterative Allocation

- Final Matrix is sum of rounds

\[
\begin{array}{ccc}
0 & 0 & 0 \\
1 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 1 \\
\end{array}
\]

\[
+ \quad \begin{array}{ccc}
0 & 1 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{array}
\rightarrow \quad \begin{array}{ccc}
0 & 1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 1 \\
\end{array}
\]
Setting Arbiter Priority

• Input and output arbiters priority can be set in different ways
• PIM – parallel iterative matching
  – Random priority
• iSLIP
  – Round-robin priority
  – Adjust only when both input and output win
Lonely Output Allocator
Wavefront Allocator
Allocator Performance

![Graph showing Allocator Performance](image)
Question of the day

• What would you rather have:
  – a perfect allocator and a switch with no speedup, or
  – an average allocator and a switch with a speedup of 2?
Allocator Summary

- **Arbiters**
  - Resolve requests for a single resource
  - Ex: channel arbitration in dropping flow control
  - Fairness: local, global

- **Allocator**
  - Match requesters to resources
  - Ex: switch allocation in an input-queued router
  - Exact solutions expensive
  - Fast approximations: separable allocators
  - Close gap with speedup