Architecture and Optimal Configuration of a Real-Time Multi-Channel Memory Controller

Manil Dev Gomony\textsuperscript{1}, Benny Akesson\textsuperscript{2} and Kees Goossens\textsuperscript{1}

\textsuperscript{1} Eindhoven University of Technology, The Netherlands
\textsuperscript{2} CISTER-ISEP Research Centre, Polytechnic Institute of Porto, Portugal
Heterogeneous multi-processor platforms

- Consumer devices mostly run on heterogeneous Multi-Processor platforms

- Dynamic Random Access Memory (DRAM) is typically used as shared main memory for cost reasons
Memory controller

- The arbiter grants memory access to one of the memory clients at a time
- Command generator issues memory commands to serve a memory request
Main memory bandwidth requirement > 10 GB/s by 2013
   - Memory power consumption scales up with memory operating frequency
   \[ \rightarrow \text{“Go parallel”} \]

Multi-channel memories
   - Each channel is an independent memory module with dedicated data and control lines
   - WIDE IO DRAM (4 channels)
We need an architecture that can support load balancing
We need to find the optimal mapping of memory clients to memory channels

Requirements
- Bandwidth
- Latency
- Communication
- Capacity
- Request sizes
Outline

• Introduction
• Problem statement
• Multi-channel memory controller architecture
• Real-time guarantees
• Mapping
• Case study
• Summary
Load balancing - past approach

- Interleave memory requests across all the memory channels available

Requirements
  - Bandwidth
  - Latency
  - Communication
  - Capacity
  - Request sizes

- Poor memory utilization!
**Our approach**

1. **Channel Interleaving**: a memory transaction can be chopped into smaller sized transactions called “Service Units” and mapped across different memory channels.

2. Each memory channel can be mapped with different numbers of service units from a single client.
The Multi-channel interleaver consists of an Atomizer, Channel selector (CS) and a Sequence generator.

- Atomizer chops the incoming requests into a number of service units of fixed size.
- CS routes the service units to the different memory channels.
Logical view of the entire memory space must be continuous to avoid explicit data partitioning and data movement while writing the application program.

\[ \text{PhyAddr}_{ch} = \frac{\text{ReqAddr} - \text{BaseAddr}}{\text{Request size} / N_{ch}} + \text{BaseAddr}_{ch} \]

Number of service units allocated to the channel.
Outline

• Introduction
• Problem statement
• Multi-channel memory controller architecture
• Real-time guarantees
• Mapping
• Case study
• Summary
Heterogeneous multi-processor platforms consist of a mix of firm real-time and soft real-time memory clients.

- **Firm real-time** → strict latency and/or bandwidth requirements on memory traffic
  - Must be guaranteed at design-time

- **Soft real-time** → average memory bandwidth requirements
  - Allocate as much bandwidth as possible to improve their average-case performance
To provide guarantees on memory bandwidth and latency to a memory client, a real-time memory controller uses

- A fixed set of memory access parameters such as burst size, page-policy etc. \(\rightarrow\) bounds transaction execution time
- An arbiter belonging to the class of *Latency-Rate (LR) Server* \(\rightarrow\) bounds response time

![Diagram of a real-time memory controller](image)
Latency-Rate (LR) servers

- **LR servers** is a general model to capture the worst-case behavior of various arbiters (servers) such as TDM, Round Robin, etc.

- The minimum service provided by the arbiter to a client depends on two parameters namely *service latency* ($\Theta$) and *allocated rate* ($\rho'$)

- Worst-case latency for a memory transaction with $N$ service units is given by

$$L^{\text{max}} = \Theta + \frac{N}{\rho'}$$
Outline

• Introduction
• Problem statement
• Multi-channel memory controller architecture
• Real-time guarantees
• Mapping
• Case study
• Summary
Optimal mapping of memory clients to memory channels depends on
1. Granularity at which the memory requests are interleaved in each channel
2. Bandwidth allocated to each memory client in each channel
The rate (bandwidth) allocated to firm real-time clients must be minimized $\rightarrow$ Soft real-time clients can be allocated with more bandwidth

Find the mapping of service units to the memory channels and an allocated rate such that the sum of allocated rates to the memory clients in all channels is minimized

$$\text{minimize : } \sum_{c \in C} \sum_{r \in R} \rho_c(r)$$

We formulated the problem as an integer programming problem
• **Constraint 1:** Meet the latency requirements
  - Service units of a transaction may get served at different time instants in different memory channels

\[
\forall c \in C, r \in R : L_{\text{max}}(r) = \max_{c \in C} L_{c,\text{max}}(r)
\]

\[
\forall c \in C, r \in R : L_{\text{max}}(r) - L_{c,\text{max}}(r) \geq 0
\]
Constraints

- **Constraint 2**: Meet the communication requirements
  - Clients that need to communicate must be mapped to the same set of memory channels and data must be aligned

- **Client 1 request (Write)**
  
  - SU 1
  - SU 2
  - SU 3
  - SU 4

- **Client 2 request (Read)**
  
  - SU 1
  - SU 2

\[ \forall c \in C, r \in R, g(r_i) = g(r_j) : N_c(r_i) \times 2^{N'_c(r_j)} = N_c(r_j) \times 2^{N'_c(r_i)} \]
Constraints

- **Constraint 3**: Meet the bandwidth requirements

\[
\forall r \in R : \sum_{c \in C} \rho_c'(r) \geq \frac{b_{min}(r)}{b_{ch}(r)}
\]

- **Constraint 4**: The bandwidth allocated to all the clients in each channel must be within the channel’s bandwidth capacity

\[
\forall c \in C : \sum_{r \in R} \rho_c'(r) \leq 1
\]

- **Constraint 5**: The memory capacity allocated to all the clients in each channel must be within the channel’s memory capacity

\[
\forall c \in C : \sum_{r \in R} \frac{N_c(r)}{q(r)} \times B_{req}(r) \leq B_{ch}(c)
\]
### Mapping overview

**Optimization problem formulation**

Objective function:

\[
\text{minimize} : \sum_{c \in C} \sum_{r \in R} \rho_c'(r)
\]

Constraints

**Memory client requirements**
- Latency
- Bandwidth
- Communication Capacity
- Request size

**Architecture specification**
- # Memory channels
- Gross bandwidth
- Service unit size

**Optimization tool (CPLEX)**

**Mapping results**
- Mapping of service units of each client to each channel
- Allocated rates for each client in each channel
We performed simulations with memory clients of different bandwidth/latency requirements and request sizes.

<table>
<thead>
<tr>
<th>Memory Channels</th>
<th>Number of Clients</th>
<th>Run-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>25</td>
<td>3 hrs</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>10 hrs</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>2 days</td>
</tr>
</tbody>
</table>

In our simulations, the tool was able to find an optimal solution within first 15 minutes!
Outline

• Introduction
• Problem statement
• Multi-channel memory controller architecture
• Real-time guarantees
• Mapping
• Case study
• Summary
Configuring a 4-channel Wide IO DRAM in a 1080p HD video processing system

- Multi-channel memory: Wide IO SDR 200 MHz (JEDEC)
We selected a service unit size of 64 Bytes considering the request sizes.

<table>
<thead>
<tr>
<th>Clients</th>
<th>Bandwidth Requirements (MB/s)</th>
<th>Latency requirements (clock cycles)</th>
<th>Request sizes (Bytes)</th>
<th>Communication group</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP&lt;sub&gt;out&lt;/sub&gt;</td>
<td>1</td>
<td>-</td>
<td>128</td>
<td>1</td>
</tr>
<tr>
<td>VE&lt;sub&gt;in&lt;/sub&gt;</td>
<td>769.8</td>
<td>-</td>
<td>128</td>
<td>1</td>
</tr>
<tr>
<td>VE&lt;sub&gt;out&lt;/sub&gt;</td>
<td>93.3</td>
<td>-</td>
<td>128</td>
<td>2</td>
</tr>
<tr>
<td>GPU&lt;sub&gt;in&lt;/sub&gt;</td>
<td>1000</td>
<td>-</td>
<td>256</td>
<td>2</td>
</tr>
<tr>
<td>GPU&lt;sub&gt;out&lt;/sub&gt;</td>
<td>500</td>
<td>102</td>
<td>256</td>
<td>3</td>
</tr>
<tr>
<td>LCD&lt;sub&gt;in&lt;/sub&gt;</td>
<td>500</td>
<td>102</td>
<td>256</td>
<td>3</td>
</tr>
<tr>
<td>CPU</td>
<td>150</td>
<td>-</td>
<td>128</td>
<td>4</td>
</tr>
</tbody>
</table>
Mapping results

<table>
<thead>
<tr>
<th>Client</th>
<th>Channel 1</th>
<th>Channel 2</th>
<th>Channel 3</th>
<th>Channel 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N</td>
<td>$\rho'$</td>
<td>N</td>
<td>$\rho'$</td>
</tr>
<tr>
<td>IP$_{out}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VE$_{in}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VE$_{out}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GPU$_{in}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>GPU$_{out}$</td>
<td>2</td>
<td>0.4</td>
<td>2</td>
<td>0.4</td>
</tr>
<tr>
<td>LCD$_{in}$</td>
<td>2</td>
<td>0.4</td>
<td>2</td>
<td>0.4</td>
</tr>
<tr>
<td>CPU</td>
<td>2</td>
<td>0.16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>6</td>
<td>0.96</td>
<td>4</td>
<td>0.8</td>
</tr>
</tbody>
</table>

- Requests are interleaved across channels
  - To meet latency requirements $\rightarrow$ GPU$_{out}$, LCD$_{in}$
  - To meet bandwidth requirements $\rightarrow$ GPU$_{in}$, VE$_{in}$
  - To meet communication requirements $\rightarrow$ VE$_{out}$ + GPU$_{in}$, VE$_{in}$ + IP$_{out}$
Outline

- Introduction
- Problem statement
- Multi-channel memory controller architecture
- Real-time guarantees
- Mapping
- Case study
- Summary
Summary

• We devised a real-time multi-channel memory controller architecture that enables interleaving memory requests across memory channels at different granularities

• We proposed an optimal integer programming based approach to configure the multi-channel memory controller for minimum bandwidth utilization
  – Bandwidth
  – Latency
  – Communication
  – Memory capacity
Questions?