Co-Synthesis of a Configurable SoC Platform based on a Network on Chip Architecture

Mário P. Véstias & Horácio C. Neto
INESC-ID (PROSYS), Lisbon, Portugal

ASP-DAC’06
Yokohama, Japan
January 24-27, 2006
Outline

1. Introduction and Motivation
2. What Exists?
3. Our project proposal
4. Target architecture
5. Design methodology
6. Application example
7. Conclusions
8. Future work
1. Introduction and Motivation

- Development of systems for high-capacity data processing on configurable platforms
- Working on architectures and design methodologies for SoC design
- Developed a platform-based methodology with IP and architecture reuse.

Problems with the interconnection architecture (bus-based):
- Does not support the efficient interconnection of many IP in a structured way.
- Increasing connection difficulties – lack of scalability
- Need reuse of interconnection!

Solution => Network on Chip
2. What exists

- There exists some work on NoC development
  - Topologies, structures, protocols, etc.
- There exists lots of work on cosynthesis and codesign, but for bus-based architectures
  - Allocation, mapping, scheduling for Hw/Sw systems
- It is time to adapt many of these techniques and ideas to develop SoC systems based on the NoC paradigm
  - Some tools already exist, but not a complete co-synthesis framework
3. Our project proposal

- In a first step, our project consisted on:
  - Develop a co-synthesis methodology based on a NoC architecture
  - Study the impact on design productivity
  - Study the impact on design complexity
  - Study the impact on design quality
4. Target architecture

- **NoC:**
  - 2D-mesh topology
  - Routing, flow-control, switching, arbitration, buffering

- **Parameterization:**
  - 2D-size
  - Type of core
  - Link size
4.1 Target architecture - behavior

NoC behavior similar to the OSI communication architecture

Layered Architecture
- Application
- Session
- Transport
- Network
- Data-Link
- Physical

Platform Components
- IP cores
- Network Interfaces
- Network Routers
- Physical links

IP Core
- Application
- Session
- Transport
- Network
- Data-Link
- Physical

NI
- Session
- Transport
- Network
- Data-Link
- Physical

Router
- Network
- Data-Link
- Physical

IP Core
- Application
- Session
- Transport
- Network
- Data-Link
- Physical
4.2 Target Architecture - Router

Routing – XY algorithm
Flow-control: 2 way handshake
Switching: store and forward
Arbitration: round-robin
Buffering: output
4.3 Target Architecture - Network interface

- Shared Memory comm
- Functions:
  - Memory management
  - Task scheduling
  - Reassembly
  - Segmentation
  - Encapsulation
  - Flow-control
4.4 Target Architecture - Packet structure

- Data packets
- Configuration packets
- Token packets

<table>
<thead>
<tr>
<th>Layer</th>
<th>Fields</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>Addr, Port, Instr/Prty</td>
<td>5, 4, 3, 24</td>
</tr>
<tr>
<td>Transport</td>
<td>Port, Instr/Prty</td>
<td>4, 3, 24</td>
</tr>
<tr>
<td>Session</td>
<td>Instr/Prty</td>
<td>3, 24</td>
</tr>
<tr>
<td>Application</td>
<td>Data</td>
<td>24</td>
</tr>
</tbody>
</table>

Data packets contain data, configuration packets contain configuration information, and token packets contain tokens for authorization or other purposes.
4.5 NoC Characterization - performance

- **Link latency** (LL) – 1 cycle;
- **Resource generation latency** (RGL) – 4 cycles;
- **Resource reception latency** (RCL) – 5 cycles;
- **Resource to resource latency** (R2RL) - …;
- **Resource to resource bandwidth** (R2RB) – f/5 packets/s.

\[
\text{Edge delay} = \frac{RGL + RCL + LL \times NR}{f(\text{working frequency})}
\]

\[
\text{Edge delay}_{\text{pipe}} = \frac{1}{R2RB} \times (NR + NP)
\]

Can forward up to 750 Mpackets/s => 6Gbps
### 4.6 NoC Characterization - area

<table>
<thead>
<tr>
<th>Block</th>
<th>Size (slices)</th>
<th>BRAM</th>
<th>% XC2V6000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router (8bits)</td>
<td>189</td>
<td>0</td>
<td>0.56</td>
</tr>
<tr>
<td>NI</td>
<td>121</td>
<td>1</td>
<td>0.36</td>
</tr>
</tbody>
</table>
5. Design Methodology

Generic NoC Platform

Architecture instance

Mapping
- Design Constraints
- Quality

Analysis
- Design Constraints
- Quality

Allocation
- Design Constraints

Applications
5.1 Design Methodology - Allocation

- Size of topology
- Type of cores

SA Algorithm
5.2 Design Methodology - Mapping

Assigns task, data transfer and variable to core, link and memory - SA Algorithm

NoC Architecture

IHDFG

Map

Analysis

Unrolled Graph

Unroll <- Unroll + 1

Improved?

yes

no

Save Solution

yes

no

End

Unroll?
5.3 Design Methodology - Analysis

Quality = $C_{\text{throughput}} + C_{\text{throughput}}$

$C_x = \begin{cases} 
K \times \frac{C_i(P)}{C_i}, & \text{without constrain} \\
K_a \times \max \left( 0, \frac{(C_i(P) - C_i)}{C_i} \right), & \text{with constrain}
\end{cases}$
6 Application Example - JPEG

JPEG algorithm for color images with block size 8 x 8

<table>
<thead>
<tr>
<th>Task</th>
<th>Size (slices)</th>
<th>BRAM</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB2YCbCr</td>
<td>204</td>
<td>0</td>
<td>64</td>
</tr>
<tr>
<td>2D-DCT</td>
<td>1612</td>
<td>1</td>
<td>168</td>
</tr>
<tr>
<td>Quantizer</td>
<td>312</td>
<td>1</td>
<td>64</td>
</tr>
<tr>
<td>Huffman</td>
<td>176</td>
<td>1</td>
<td>192</td>
</tr>
</tbody>
</table>
6.1 Application Example - Results

The solution processes two blocks of \(8\times 8\) \(\times 24\) bits in 3.8 \(\mu\)s => 800 Mbps (VirtexII XC2V6000)

<table>
<thead>
<tr>
<th>Image size</th>
<th>HW NoC sec/fps</th>
<th>Pentium 4 at 1.7 GHz</th>
<th>HW bus sec/fps</th>
</tr>
</thead>
<tbody>
<tr>
<td>640(\times)480</td>
<td>0.009 (108)</td>
<td>0.046</td>
<td>0.055</td>
</tr>
<tr>
<td>800(\times)600</td>
<td>0.015 (67)</td>
<td>0.071</td>
<td>0.086</td>
</tr>
<tr>
<td>1024(\times)768</td>
<td>0.024 (42)</td>
<td>0.110</td>
<td>0.14</td>
</tr>
</tbody>
</table>
7 Conclusions

- Design complexity – easier to design due to scalability, efficient interconnection of IP and communication/performance “independence”
- Design productivity – design methodology accomplishes goals more efficiently
- Design quality – better for communication intensive applications. Better throughput; latency(?). Hybrid structure with routers and buses must be considered
7 Conclusions

- A research over a set of IP cores concluded that smaller routers must be implemented to improve cost, power consumption and performance;
- Many NoC parameters must be part of the design methodology to improve final system quality – buffer size, switching capacity, arbitration policy, topology, etc.
8 Future work

- Considering a more generic topology with routers consisting of more than 1 local port and sharing of router resources to improve cost, performance and power dissipation;
- Increase the set of configurable parameters of the NoC architecture in the design methodology;
Co-Synthesis of a Configurable SoC Platform based on a Network on Chip Architecture

Mário P. Véstias & Horácio C. Neto
INESC-ID (PROSYS), Lisbon, Portugal

ASP-DAC’06
Yokohama, Japan
January 24-27, 2006