Using partial reconfiguration in an embedded message-passing system

FPGA Seminar @ UofT
March 2011

Manuel Saldaña, Arun Patel
ArchES Computing

Hao Jun Liu, Paul Chow
University of Toronto
ArchES-MPI framework

- Co-Simulation
- Profiling
- Partial Reconfiguration
- MPI-to-HDL
- NoC infrastructure and Portability
- Libraries and applications
Contents

- Motivation for doing research in Partial Reconfiguration (PR)
- PR and the message passing framework
- Vector Engine example
- Conclusions
Why partial reconfiguration?

- Improved Xilinx PR flow
Why partial reconfiguration?

- Improved Xilinx PR flow
- Largest Virtex7 FPGA
  - 1.2 M LUTs, 2.4 M FF, 1292 BRAMs, bitstream size 54 MB
- Multi-FPGA devices (Xilinx Stacked Silicon Interconnect Technology)

Typical MPI

MPI processes

Cluster or Supercomputer
ArchES-MPI

MPI processes

Heterogeneous Computing Elements

Hardware Accelerator A

Hardware Accelerator A

MicroBlaze

ArchES-MPI
PR and ArchES-MPI

Hardware Reuse

ArchES-MPI

Hardware Accelerator A

Hardware Accelerator A
PR and ArchES-MPI

Hardware Reuse

ArchES-MPI

Hardware Accelerator A

Hardware Accelerator B

MicroBlaze

Intel Xeon Inside

Intel Xeon Inside

R0

R1

R2

R3

R4
PR and ArchES-MPI

Easy parallel application launch (not yet implemented)

- Typical MPI
  
  mpiexec -np 2 foo.elf : -np 3 bar.elf
PR and ArchES-MPI

Easy parallel application launch  (not yet implemented)

- Typical MPI
  mpiexec -np 2 foo.elf : -np 3 bar.elf

- ArchES-MPI
  mpiexec -np 2 foo.elf : -np 3 bar.bit

.elf ~ .bit  (binary executable files)
PR and ArchES-MPI

Dynamic Process Creation  (not yet implemented)

MPI_Comm_spawn(...);
PR and ArchES-MPI

Dynamic Process Creation (not yet implemented)

MPI_Comm_spawn(...);
Template-based bitstreams

A library of pre-built bitstreams for FPGAs ...

NoC

Reconfigurable Module (RM)

Dynamic Region

Static Region
Template-based bitstreams

Code is executed by a Computing Element

- executable.elf → CPU
- executable.bit → Dynamic Region (FPGA)
Template-based bitstreams

PlanAhead snapshots of FPGA layouts

1 Dynamic Region

4 Dynamic Regions

8 Dynamic Regions
Template-based bitstreams

One dynamic region can be as large as an FPGA

~XC5VLX30  XC7VLX2000TFHG

<table>
<thead>
<tr>
<th></th>
<th>XC5VLX30</th>
<th>Dynamic PR Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUTs</td>
<td>19200</td>
<td>19200</td>
</tr>
<tr>
<td>FF</td>
<td>19200</td>
<td>38400</td>
</tr>
<tr>
<td>BRAM</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>DSPs</td>
<td>32</td>
<td>64</td>
</tr>
</tbody>
</table>
Template-based bitstreams

35 dynamic regions (~XC5LX30) in one XC7VLX2000TFHG
Logic in static and dynamic regions

ArchES NoC

Infrastructure to support a message passing model

FPGA

FSB/PCIe Interface

LVDS interface

MGT Interface

FPFGA Accelerator

MPE

Embedded Processor

MEM

DDR

Packet

Hardware Accelerator

Hardware Accelerator

Embedded Processor

MPI

MEM

CLK Generator

Routing tables

Each line is two FIFOs (one in each direction)
The Message-Passing Engine (MPE)

Main Compute Pipeline

Ctrl FSM

Data Fifo

Cmd Fifo

Wrapper

Status and Control signals (Reset, busy, done, enable)

Network-on-Chip

R0

R1

R2

R3

R4
Generic vs specific templates

Network-on-Chip

MPE

<table>
<thead>
<tr>
<th>Static Region</th>
<th>Dynamic Region</th>
</tr>
</thead>
</table>

Cmd Fifo

Data Fifo

Wrapper

Ctrl FSM

Main Compute Pipeline

Status and Control signals (Reset, busy, done, enable)

PR Flag

FF

SRL

gRst

Generic

NoC
Generic vs specific templates

Network-on-Chip

Static Region

Dynamic Region

MPE

Wrapper

Ctrl FSM

Main Compute Pipeline

Data Fifo

Cmd Fifo

Status and Control signals (Reset, busy, done, enable)

Application-specific

NoC
PR Synchronization, Data Store and Restore

Configuration Controller
X86
RM_A
RM_B

Partial Reconfiguration Process

Processor-Initiated PR

Time
PR Synchronization, Data Store and Restore

Configuration Controller X86
RM_A
RM_B

Partial Reconfiguration Process

Processor-Initiated PR

RM-Initiated PR

Configuration Controller X86
RM_A
RM_B

Partial Reconfiguration Process

Time
Ease of use and Portability

User Application (X86)

ArchES-MPI SW

FSB/PCIe

ICAP

ArchES-MPI

ShMem

Bridge

NoC

SelectMap

FPGA 0

FPGA 1

Board 0

ARCHES_MPI_Reconfig (bar.bit, boardID, fpgaID);
Simple Vector Engine Example

Reconfigurable Modules

RM_MUL
RM_SUB
RM_ADD

Network-on-Chip

MPE

Cmd Fifo

Data Fifo

Wrapper

RR_0
RR_1
RR_2
RR_3

Vector Elements

RR_2
RR_1
RR_0
RR_3
RR_2
RR_1
RR_0

Dynamic Region
Static Region
Results

- 100 Load-Accumulate-Store cycles
- Vectors of 4000 floating point numbers
- Partial bitstream size of 130872 bytes

<table>
<thead>
<tr>
<th>Num of PR events</th>
<th>PCIe</th>
<th></th>
<th>FSB</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>18.695</td>
<td>425</td>
<td>6.466</td>
<td>308</td>
</tr>
<tr>
<td>10</td>
<td>1.912</td>
<td>43</td>
<td>0.666</td>
<td>32</td>
</tr>
<tr>
<td>1</td>
<td>0.231</td>
<td>5</td>
<td>0.085</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0.004</td>
<td>-</td>
<td>0.021</td>
<td>-</td>
</tr>
</tbody>
</table>
PAR times

95 min / 4 bitstreams = 25 min per bitstream
Conclusions

- developed a working framework that people can use to implement applications with PR capability
- the framework adds a portability layer across different platforms (FSB or PCIe)
- two ways to trigger PR
  - processor-initiated PR
  - hardware-initiated PR
- messages can be used to store and restore data
- the same generic, template-based bitstreams can be re-used for different applications
- template bitstreams may reduce the development time: focus only on the application engine
- bring high-level abstractions into the FPGA domain
Thank you !
MPI_Init();  // <--- Initial bitstream configuration (Template and initial PR bitstreams)
...
MPI_Send ( ..., dest, CFG_TAG,...);

MPI_Recv ( status_data_RM_A, … , dest, OK_TO_CFG_TAG, ...);
ARCHES_MPI_Reconfig ( RM_B.bit, board_num, fpga_num );
MPI_Send ( status_data_RM_B, … , dest, CFG_DONE_TAG);
...

MPI_Recv ( status_data_RM_A, … , dest, REQ_TO_CFG_TAG, ...);
ARCHES_MPI_Reconfig ( RM_B.bit, board_num, fpga_num );
MPI_Send ( status_data_RM_B, … , dest, CFG_DONE_TAG);
...

Processor-initiated PR
RM-initiated PR
Supported Functions

**Misc. Functions:**
- MPI_Init
- MPI_Finalize
- MPI_Comm_rank
- MPI_Comm_size
- MPI_Wtime
- MPI_Get_count

**Point-to-Point:**
- **Blocking**
  - MPI_Send*
  - MPI_Rsend*
  - MPI_Recv*
- **Non-blocking**
  - MPI_Isend
  - MPI_Issend
  - MPI_Irecv
  - MPI_Test
  - MPI_Wait
  - MPI_Waitall

**Collective Operations:**
- MPI_Barrier
- MPI_Broadcast
- MPI_Reduce
- MPI_Allreduce
- MPI_Gather

**MPI-2:**
- MPI_Alloc_mem
- MPI_Free_mem

* MPE functionality
Debugging: Multi-FPGA System-Level Simulation

- Test SW and HW ranks all at once
- No need to resynthesize
- Full visibility into the FPGA
- Good for modeling application-defined protocols at initial stages of development

ACP2
ACP1
ACP0

ModelSim
ModelSim
ModelSim

Intel Quad-core Xeon

MPI Messages

FPGA signals

Stdout
Profiling: Jumpshot
(Daniel Nunes @ UofT)

- Well-known tool
- Extracts MPI protocol states from the MPE
- Profile just like in Software
- Works only for embedded processors and hardware engines
Template-based bitstreams

One dynamic region can be as big as an FPGA

<table>
<thead>
<tr>
<th></th>
<th>XC5VLX30</th>
<th>Dynamic PR Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUTs</td>
<td>19200</td>
<td>19200</td>
</tr>
<tr>
<td>FF</td>
<td>19200</td>
<td>38400</td>
</tr>
<tr>
<td>BRAM</td>
<td>32</td>
<td>36</td>
</tr>
<tr>
<td>DSPs</td>
<td>32</td>
<td>48</td>
</tr>
</tbody>
</table>