Dynamic Reconfiguration of Xilinx FPGAs

Enhanced Architectures, Design Methodologies, & CAD Tools

Patrick Lysaght,
Xilinx Research Labs
Acknowledgements

• Co-authors
  – Jay Young, Jeff Mason, Brandon Blodget & Brendan Bridgford

• Xilinx contributors
  – Eric Shiflet, Jason Moore, Manuel Uhm, Steve Lass, Adam Donlin & many more

• Prof Juergen Becker’s team for the “bus macros”
  – Michael Huebner, Michael Ullmann & Tobias Becker at the University of Karlsruhe, Germany
Overview

- Motivation
- New V4 Architectural enhancements
- Reconfiguration latencies
- Design flow for simple audio demonstrator
- Partially reconfigurable regions and modules
- Static-dynamic region interfaces
- Early-access PR design flow
Motivation

• End: to enable dynamic reconfiguration
  – Convert from heroic activity to reliable design process

• Combination of means:
  – Architectural support for faster reconfiguration & finer granularity of dynamic reconfiguration
  – Flexible, high-bandwidth, high-availability interfaces between static and dynamic regions
  – New design methodology and CAD tool support
Finer Reconfiguration Granularity

Static Region

Reconfigurable Region

Static Region
Finer Reconfiguration Granularity

Reconfigure fabric in units of 16 CLB rows

Static Region
Reconfigurable Region
Static Region
Finer Reconfiguration Granularity

Reconfigure fabric in units of 16 CLB rows

Constant 1,312 bits per reconfig. frame
Finer Reconfiguration Granularity

- Reconfigure fabric in units of 16 CLB rows
- Constant 1,312 bits per reconfig. frame
- Fast reconfiguration with 32-bit wide 100 MHz V-4 ICAP
Lower Reconfiguration Latencies

Reconfiguration Latency (ms)

- 2VP2
- 2VP4
- 2VP7
- 2VP20
- 2VPX20
- 2VP30
- 2VP40
- 2VP50
- 2VP70
- 2VPX70
- 2VP100
- V4LX15
- V4LX25
- V4LX40
- V4LX60
- V4LX80
- V4LX100
- V4LX160
- V4LX200

% of device reconfigured:
- 10%
- 25%
- 50%
- 75%
- 100%
Lower Reconfiguration Latencies

Reconfiguration Latency (ms)

~ Order of magnitude improvement in latency when reconfiguring 25% of a XC2VP100 versus 25% of a XCV4LX100

(Both devices have approximately 100,000 logic elements each)
Multiple Reconfiguration Regions

Reconfigurable modules can be located in the same column.
Multiple Reconfiguration Regions

Reconfigurable modules can be located in the same column.
Multiple Reconfiguration Regions

Reconfigurable modules can be located in the same column

No restrictions on SRL16s and LUT RAM outside 16-CLB range of dynamic regions
Virtex-II and Virtex-II Pro

Arbitrary rectangular reconfiguration region now supported

Software automatically excludes LUTRAM and SRL16 constructs from fixed regions in reconfiguration columns
New Interface: Bus Macros

Preassigned LUT

Interface to Static region

Interface to Reconfigurable region

Preassigned LUT

Standard Slice Macro

High Density Layout of Slice Macros

Based on original work by Prof. Juergen Becker’s team at the University of Karlsruhe
Partial Reconfiguration Region (PRR)

XCV4LX25

Static (Base) Design
Partial Reconfiguration Region (PRR)

The placer will not locate any logic elements of the base design in a PRR.
Partial Reconfiguration Region (PRR)

XCV4LX25

The placer will not locate any logic elements of the base design in a PRR.

But the router can route static connections through one or more PRRs.
Partial Reconfiguration Region (PRR)

**Partial Reconfiguration Region (PRR)**

**XCV4LX25**

The placer will not locate any logic elements of the base design in a PRR.

But the router can route static connections through one or more PRRs.

These static routes in the PRR are not affected when PRMs are reconfigured into the PRR.
Partial Reconfiguration Modules (PRMs)
Partial Reconfiguration Modules (PRMs)

All PR module logic and routing are restricted to the PR region
Directional Bus Macro Interfaces

High-Pass Filter

PRM

PRR

Static Design
Directional Bus Macro Interfaces

PRM

High-Pass Filter

R2L: Audio in

PRR

Static Design
Directional Bus Macro Interfaces

L2R: Audio out

R2L: Audio in

High-Pass Filter

PRM

Static Design

PRR
New PR Design Flow

Reconfigurable Design Specification

Design Partitioning

Design Floorplanning and Budgeting

Top level implementation

Static Design implementation

PRM implementation

Merged Bitstream Generation

PRMs: HDL Files

Static and Toplevel Design HDL Files

Placement Constraints

Placement and Context Constraints

Implemented Static Design

Implemented PRMs

Static Routes Excluded from PRMs

Static Design Bitstreams Merged with PRM Bitstreams

PRM Bitstreams

PRR ‘Blank’ Bitstreams

Static Design

Static Routes

Excluded from PRMs
New PR Design Flow

Reconfigurable Design Specification

Design Partitioning

Design Floorplanning and Budgeting

Top level implementation

Static Design implementation

PRM implementation

Merged Bitstream Generation

Implemented Static Design

Implemented PRMs

Static and Toplevel Design HDL Files

Static Routes Excluded from PRMs

Placement Constraints

Placement and Context Constraints

PRMs: HDL Files

PRM Bitstreams

PRR ‘Blank’ Bitstreams

Static Design Bitstreams Merged with PRM Bitstreams
New PR Design Flow

1. **Design Partitioning**
2. **Design Floorplanning and Budgeting**
3. **Top level implementation**
   - PRMs:
     - HDL Files
     - Placement Constraints
     - Placement and Context Constraints
4. **Static Design implementation**
   - Implemented Static Design
   - Placement and Context Constraints
5. **PRM implementation**
   - Implemented PRMs
   - Static Routes Excluded from PRMs
6. **Merged Bitstream Generation**
   - PRM Bitstreams
   - PRR 'Blank' Bitstreams
   - Static Design Bitstreams Merged with PRM Bitstreams

Reconfigurable Design Specification
New PR Design Flow

1. Design Partitioning
2. Design Floorplanning and Budgeting
3. Top level implementation
4. Static Design implementation
5. PRM implementation
6. Merged Bitstream Generation

- Reconfigurable Design Specification
- Static and Toplevel Design HDL Files
- Placement Constraints
- Placement and Context Constraints
- PRMs: HDL Files
- Implemented Static Design
- Implemented PRMs
- Static Routes Excluded from PRMs
- PRM Bitstreams
- PRR ‘Blank’ Bitstreams
- Static Design Bitstreams Merged with PRM Bitstreams
Design Flow Highlights

- Recognize mutually exclusive functionality
- Partition into static and dynamic components
- Determine PR regions & associated PR modules
- Decide Bus Macros interfaces
- Create top-level design context with:
  - All globals such as IO ports, clock constructs (DCMs, BUFGs), black-boxes instantiations for PR regions, all Bus Macros, signals, etc
- Create prescribed HDL file hierarchy
Recommended Directory Structure

- **base:**
  - contains implementation of static design

- **merges:**
  - PRMs are merged with base design

- **reconfigmodules:**
  - separate implementation of each PRM

- **synth:**
  - HDL for top level, base design and each PRM
    - synthesized in unique directories

- **top:**
  - top-level netlist and design constraints file (UCF)
    - are placed in this separate directory
PlanAhead:
Hierarchical Design & Analysis Tool

- Deploy between synthesis & PAR
- Graphically correlate logic and physical design views
- Estimate resource requirements, timing and bitstream sizes before place & route
- Determine size, shape and placement of PRRS
- Determine location of Bus Macros
- Run multiple design rule checks
- Output constraints files
Context, Base & PRM Implementation

• Context (top-level design) implemented first
• Base design implemented with top-level context “top.ucf” file
  – All static resources are constrained into one AREA_GROUP
  – Outputs “static.used” file detailing routing consumed in PRRs
• Each PRM is implemented in turn
  – “static.used” file renamed as “arcs.exclude” file and used with “top.ucf” for each PRM
  – PRM constrained into AREA_GROUPs with resource RANGE constraints and MODE = RECONFIG
Context, Base & PRM Implementation

Extract from file “Reconfigmodules\bandpass\top.ucf”

“# area group for static
INST "ac97_if_inst" AREA_GROUP = "AG_static_system" ;
INST “audio_base_design” AREA_GROUP = "AG_static_system" ;

# area group for reconf module
INST "inst_reconfig_filter" AREA_GROUP = "AG_inst_reconfig_filter" ;
AREA_GROUP "AG_inst_reconfig_filter" RANGE = SLICE_X0Y64:SLICE_X21Y191 ;
AREA_GROUP "AG_inst_reconfig_filter" RANGE = DSP48_X0Y16:DSP48_X0Y47 ;
AREA_GROUP "AG_inst_reconfig_filter" RANGE = RAMB16_X0Y8:RAMB16_X0Y23 ;
AREA_GROUP "AG_inst_reconfig_filter" MODE = RECONFIG;”
Merge Stage

- Merge every PRM in turn with the static design
- Verify PRMs
  - Base design to PRM interface correctness
- Create the partial bitstream for each PRM
  - Dual computation of every partial bitstream
- Create the full bitstream for FPGA power-up
- Create arbitrary full bitstream for timing analysis
- Create PRM blanking bitstreams
Future Work: Early Access PR Lounge

http://www.xilinx.com/xlnx/xil_entry2.jsp?sMode=login&group=prealounge
Thank You