COMMA: A Communications Methodology for Dynamic Module Reconfiguration in FPGAs (Extended Abstract)

Shannon Koh and Oliver Diessel
School of Computer Science & Engineering, The University of New South Wales
Embedded Real-Time, and Operating Systems (ERTOS) Program, National ICT Australia, Sydney, Australia

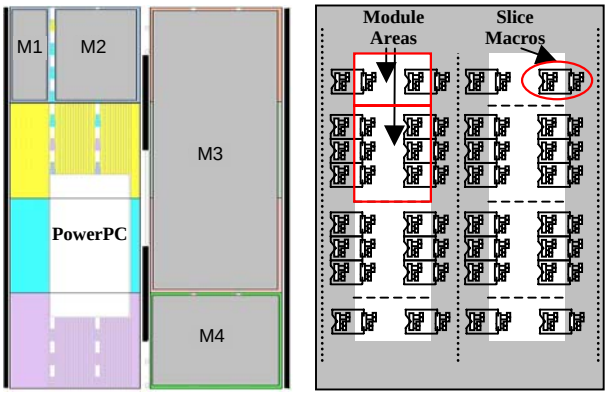
1 Introduction

On-going improvements in the scaling of FPGA device sizes and time-to-market pressures encourage the use of module-oriented design flows [3], while economic factors favour the reuse of smaller devices for high performance computational tasks. One of the core problems in proposing dynamic modular reconfiguration approaches is supporting the differing communications needs of the sequence of modules configured over time [2].

Proposals to date have not focussed on communications issues. Moreover, they have advocated the use of specific protocols [4], or they cannot be readily implemented [1], or they suffer from high overheads [5], or rely upon deprecated features such as tri-state lines [7]. In contrast, we propose a methodology for the rapid deployment of a communications infrastructure that provides the wires required by dynamic modules and allows users to implement the protocols they want. Our aim is to support new tiled dynamically reconfigurable architectures such as Virtex-4, as well as mature device families.

2 The COMMA Approach

We propose developing a set of tools to automatically generate a wiring infrastructure that is optimised for a given hardware platform and for the application requirements known at design time.



1(a): V4FX12 Pages *1(b) Infrastructure Layout*

Figure 1: Paging and Infrastructure

We target the new Xilinx Virtex-4 family of FPGAs to demonstrate our approach. This architecture is unique in that the configuration frames have a fixed-length of 41 quad-byte words, each spanning 16 CLB rows. These are tiled about the device into independently configurable and independently clocked "pages" that span half the width of the device, thereby allowing us to readily place modules as shown in Figure 1(a). External I/O banks are located towards the left and right edges of the device, but not necessarily right on the edge as in predecessor families. An additional bank is located in the centre columns of the device.

To support independent reconfiguration, we propose accommodating one reconfigurable module per page. However, a module may be placed in two or more adjacent pages such that larger ones can be accommodated, and it is also possible to divide a page into sub-pages spanning fewer CLB columns. Figure 1(a) shows modules M1 and M2 being placed into a sub-divided page, M3 into three aggregated pages and M4 into a page of its own. The above concepts can also be implemented on predecessor device families if the entire device is viewed as a single page and modules are placed into horizontally divided sub-pages.

In order to support dynamic module replacement and relocation, our approach allows virtualisation of any I/O pin in the communications infrastructure and relies upon mapping virtual to physical pins at reconfiguration time. Pin virtualisation is implemented by covering external I/O pins with the infrastructure, providing module I/O via slice macros [6], statically configuring alternative routes between module and external I/O pins, and defining Reconfigurable Data Ports (RDPs, see Figure 2) that map logical module ports to physical slice macros. At reconfiguration time, RDPs are dynamically connected either to RDPs of other modules or to external I/O pins by selecting the actual routes from those configured at application startup. RDPs are implemented as module adapters or wrappers.

Figure 1(b) illustrates a double ring layout of the infrastructure for an XC4VFX12 device that provides for 8 reconfigurable module areas (of 3 different types) and illustrates the presence of slice macros on the boundary of the grey wiring area and the white module areas. Note that any layout that envelops the communicating module and I/O pins, satisfies bandwidth requirements, and makes effective use of wiring resources can be chosen.

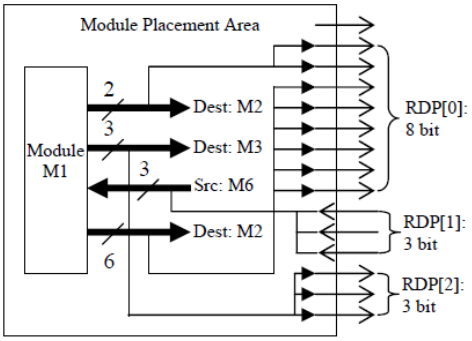


Figure 2: RDP Mappings

Communication requirements are derived from design time knowledge about the runtime sequence of module configurations, their interfaces and interconnection patterns. User constraints are also used to guide the optimisation of the inter-module wiring layout. Dynamic module placement and communication channel setup, possibly involving route selection, is performed either by on- or off-chip agents.

3 Current Development

A prototype has been developed that involves a two-stage tool flow to implement an instance of our communications architecture for a given dynamic application. In the first stage, we use Xilinx-supplied device information with user-supplied module interface, schedule and placement constraints to generate an optimised communications infrastructure consisting of HDL, constraints, slice macros and accessory macros. During the second (module finalization) stage, the infrastructure's design is used to wrap and physically implement dynamic modules on the fabric.

We have implemented a circuit-switched, multiplexer-based prototype on an XC4VFX12 device as shown in Figure 3. The infrastructure provides 4 module areas, each having 8 inputs and 8 outputs, and supports 16 external I/O pins in total. The host control switches slice macros and external I/Os with a small on-chip programmable controller that sets multiplexer selectors to direct pin-to-pin communications.

Due to the lack of support for Virtex-4 partial reconfiguration in ISE 7.1i, internal module routing ignores area constraints (Figure 3: left indicates area constraints, right indicates final routes). Our module finalisation flow solves this by integrating the infrastructure as a hard macro, preventing modules from utilising the infrastructure routes.

The maximum clock speed of the communications infrastructure after place and route, as per the tool flow above, is 358 MHz (speed grade –10, i.e. the slowest available), which should be sufficient for most modules. The infrastructure takes up 573 slices (~10.5% of the smallest Virtex-4 device, i.e. the XC4VFX12), and each module area is 576 slices large. For reference, a DES core uses 476 slices at 182 MHz and a MicroBlaze processor 988 slices at 200 MHz. The trident layout scales to 22 dynamic module areas, each of 3200 slices, for the largest Virtex-4 LX device. Reconfiguration overheads at this module size might encourage further module area subdivision.

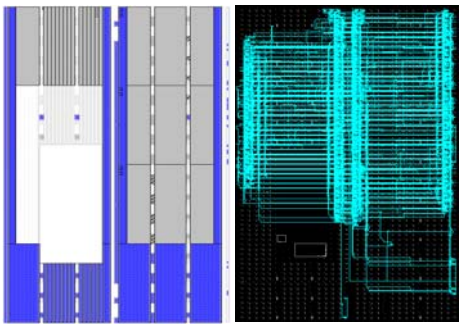


Figure 3: Implementation Constraints and Routes

We intend automating the tool flow and determining methods for optimising the routing and switching architecture in the future.

Acknowledgements: National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

[1] C. Bobda, A. Ahmadinia, M. Majer, J. Teich, S. Fekete, and J.v.d. Veen. DYNOC: A dynamic infrastructure for communication in dynamically reconfigurable devices. In FPL 2005, pp. 153 – 158.
[2] G. Brebner. The swappable logic unit: a paradigm for virtual hardware. In FCCM97, pp. 77-86.
[3] L. Chaouat, S. Garin, A. Vachoux, and D. Mlynek. Rapid prototyping of hardware systems via model reuse. In 8th IEEE Intl Workshop on Rapid System Prototyping, 1997, pp. 150-156.
[4] H. Kalte, M. Porrmann, and U. Ruckert. System-on-programmable-chip approach enabling online fine-grained 1D-placement. In IPDPS 2004, p. 141.
[5] T. Marescaux, A. Bartic, D. Verkest, S. Vernalde, and R. Lauwereins. Interconnection networks enable fine-grain dynamic multi-tasking on FPGAs. In FPL 2002, 795 – 805.
[6] P. Sedcole, B. Blodget, J. Anderson, P. Lysaght, and T. Becker. Modular partial reconfiguration in Virtex FPGAs. In FPL 2005, pp. 211 - 216.
[7] Xilinx Inc. Two flows for partial reconfiguration: module based or difference based. Xilinx App. Note 290 Sep., 2004.