Fine-grained Partial Runtime Reconfiguration on Virtex-5 FPGAs

Dirk Koch, Christian Beckhoff, and Jim Torrison
Department of Informatics, University of Oslo, Norway
Email: {dirk, christian}@recobus.de, jimtoer@ifi.uio.no

Abstract—The architecture of Xilinx FPGAs, has changed remarkable with respect to their ability to implement runtime reconfigurable systems throughout the last generations. This paper will discuss these changes and reveal an on-FPGA communication architecture that is especially tailored to Xilinx Virtex-5 FPGAs. With this architecture, modules can be integrated in a two-dimensional grid with more than a hundred of individual tiles while allowing a throughput of several GB/s to reconfigurable modules.

I. INTRODUCTION

When fitting partially reconfigurable modules into predefined regions on an FPGA, not all modules will perfectly fit into this regions and some resources might be left unused. This phenomenon is known as *internal fragmentation*. In [1], we demonstrated that runtime reconfigurable systems are very vulnerable to internal fragmentation and that the best solution to this problem is a fine tile-grid for hosting reconfigurable modules. The challenge in implementing a very fine grained tile grid is to provide communication at high throughput and at low implementation cost, while keeping the placement flexibility of that grid. As one solution to this problem, we proposed I/O bars for carrying out the communication among the reconfigurable modules. Figure 1 depicts, examples of I/O bars that consist of wiring bundles that are homogeneously routed among the reconfigurable tiles (slots) of the system. A system may provide multiple I/O bars, and modules may access or bypass bars selectively.

In the following, we will reveal changes between the Virtex-II and Virtex-5 FPGA families with respect to an implementation of a regular structured communication architectures. Next, in Section III, a two-dimensional circuit switching architecture that is based on I/O bars and tailored to Virtex-5 FPGAs will be revealed in Section IV. After this, the ReCoBus-Builder framework and the corresponding design flow for implementing runtime reconfigurable systems will be introduced in Section V.

II. DEVELOPMENT TRENDS IN FPGA ARCHITECTURE

The benefit of utilizing latest silicon processes technology for FPGAs can be used to enhance the logic density on a die, reducing power consumption, and for reducing latency, hence, allowing higher clock rates. The next paragraphs examine changes to the architecture between Xilinx Virtex-II and Virtex-5 FPGAs with respect to an implementation of a runtime reconfigurable system.

A. Changes to the Wire Architecture

The routing fabric of for Xilinx Virtex-II devices allowed to implement modules with a very high logic utilization. In Virtex-5, focus is put on enhancing the total utilization of the die. As the routing fabric takes much more area on the FPGA then actually the logic within a CLB, it is important to well balance the FPGA architecture between the logic and routing density. This development trend can be observed at many points. For example, the relatively amount of data spent for LUT values within the configuration bitstream of

a single CLB raised from 7.3% in Virtex-II to now 22.2% in Virtex-5 FPGAs. Note that a reconfiguration bit corresponds to area on the die.

While the number of local wires that start in each CLB has been kept roughly the same throughout the last Virtex generations, the number of endpoints has decreased from 241 endpoints for Virtex-II to 202 in Virtex-5 (see Fig. 2). Opposed to presentations from the vendor Xilinx [2], Figure 2 emphasizes not which CLBs can be reached within a certain amount of hops, but how wide and how long the routing channels between the surrounding CLBs are. It can be seen that channels have become thinner, shorter, and more spread over the fabric.

B. Trends in Switch Matrix Development

An important fact that is hidden in Figure 2 is that wires on Virtex-5 FPGAs cannot be homogeneously extended within a switch matrix as it is the case for most wires on Virtex-II FPGAs. This means for Virtex-5 that when starting a routing track with a dedicated wire resource in one Virtex-5 CLB, the path cannot be directly continued towards the same direction using the same corresponding dedicated wire resource in the entire destination CLB. This makes it more difficult when circuits have to be regularly mapped to a given FPGA architecture.

C. Changes in the FPGA Fabric Layout

Xilinx FPGAs provide distributed memory that allows to use look-up tables as memory elements. These can be – alternatively to the normal look-up table mode – either configured as shift registers or as tiny RAM blocks. However, while each LUT on a Virtex-II provides the distributed memory option, this feature is only available in one fourths of all look-up tables on a Virtex-5 FPGA. These special LUTs are physically located in every second row of CLB columns. This results in restrictions when distributed memory is used by reconfigurable modules that should be placed to different positions at runtime. Note that the configuration data of a logic only column is compatible to a column that is providing the distributed memory option. This can be used to remove the placement restrictions that stem from the distributed memory feature on Virtex-5. This approach has similarities to [3], where relocatability for modules that possess special resource columns was enhanced by only

Figure 1. I/O bars for integrating various reconfigurable modules.

Figure 2. Local interconnect network of different FPGA architectures. The small boxes illustrate the switch matrices of the configurable logic blocks (CLB) that combine the logic and routing resources of the elementary building blocks of Xilinx FPGAs. The figures highlight all local routing wires that leave a CLB in the center. The numbers in the surrounding CLBs denote the amount of wires that can be accessed in a particular destination.

using the switch matrices of these resources on Xilinx Virtex-4 FPGAs.

While in Xilinx Virtex-II FPGAs there exists only one class of special resource columns that provides dedicated multipliers and larger RAM blocks, these resources are available in separate types of columns for dedicated multipliers and memory in the case of a Virtex-5 device. Also the positions of these special resource columns are located more irregular in the case of Virtex-5.

While for Virtex-II, all I/O pins are located at the edges around the die, I/O pins are embedded into the fabric as special resource columns in the case of the Virtex-5 architecture. This has to be considered when floorplanning a reconfigurable system.

D. Impact of FPGA Architecture Changes on I/O Bars

The here presented discussion on the differences between Virtex-II and Virtex-5 state just a qualitative view, but point out future trends for FPGA architectures. This includes higher ratio of logic to routing resources, less connections between the basic building blocks, and a more irregular wire structure. The impact and challenges on the design and implementation of a regular structured communication architecture for runtime reconfigurable systems with respect to changes to the FPGA architecture from Virtex-II to Virtex-5 can be summarized as follows:

- *Larger LUTs and slices* require higher effort for achieving good logic utilization and influence to implement the communication architecture more compact.
- *Less wire resources* (especially along the main horizontal directions) demands to spread the architecture over multiple CLBs.
- *Missing direct extension of routing tracks within the CLBs* requires more complex routing schemes.
- *With a lower amount of endpoints*, multicast bus signals (e.g., an enable signal) will be more challenging to implement.
- *The irregular placement of dedicated resources* requires more considerations in the system design (PCB and FPGA floorplanning).

Figure 3. Regular signal routing scheme that can be applied to Virtex-5 FPGAs. The routing is recurring after two CLBs.

III. DESIGN OF A 2D CIRCUIT SWITCHING NETWORK

As mentioned in the previous section, the switch matrices of Virtex-5 devices cannot be configured for directly extending a routing path, when using the same routing wire in each CLB. However, alternative paths have been found that provide a regular signal routing scheme by swapping between two or more wire resources, as illustrated in Figure 3. In that example, the regular routing scheme is repeated after two CLBs. As each CLB provides both wire resources, from which only one is used for a particular signal path within one CLB, a second path must exist that can be interleaved within the first one. This path would use the respectively remaining wire resources, as highlighted in Figure 3.

At a first glance, it seems obvious to arrange resource slots to be two CLB columns wide, as this would allow to access the I/O bar signals at the same relative position within each slot. However, this holds only as long as no special resource columns are located within the reconfigurable region. As can be seen in Figure 3, a special resource column swaps once the regular modulo two routing scheme. This cannot be easily compensated by defining one further CLB column to be attached to the special resource column that would then correct the scheme by swapping it a second time. The reason for this is that not only the I/O bar routing has to be regularly arranged, but also the distributed memory resources.

These limitations will be circumvented, if the I/O bar would allow to connect a module at any CLB column along the bar. As all signals of an I/O bar are accessible in each CLB column for read as well as write operation, it is possible to realign the routing scheme arbitrary in each module connection point. As depicted in Figure 4, this can be implemented by using LUTs in routethrough mode. The idea of that approach is to multiplex the two interleaved signal paths of the I/O bar such that a module connection point behaves to the routing scheme of the bar like a bypassing slot that would be located at the same position.

A. Two-dimensional Extension

The here presented I/O bar concept allows to tile the reconfigurable area into resource slots that are just one CLB

Figure 4. I/O Bar with two connection points via alignment multiplexers.

Figure 5. Two-dimensional circuit switching network using I/O bars. Multiplexers in the static part perform the vertical routing while I/O bars carry out the routing in horizontal direction.

column wide and arranged in a one-dimensional manner. When following the native frame size clustering of Virtex-5 FPGAs, a slot provides 20 CLBs or 160 6-input LUTs.

For extending the concept towards a communication architecture allowing two-dimensional placement, multiple I/O bars can be instantiated one above the other as illustrated in Figure 5. In this approach, vertical routing is carried out with the help of multiplexers that are located in the static part of the system. Vertical routing may be alternatively arranged by modules that are more than one row of resource slots height. Note that each I/O bar has been designed with a forward (towards the right) and a backward signal path (towards the left) and that a module can connect to both paths. This removes communication related placement restrictions and allows to communicate between two modules regardless if the second module is located left or right beside the first module. As can be seen in Fig. 5, different topologies, including multicast topologies, can be set.

B. Implementation Cost

The implementation cost of an on-FPGA communication architecture has to be rated for wire as well as for logic/flip-flop resources that are provided by an FPGA fabric.

The smallest unit of logic that can be allocated to a reconfigurable module or an I/O bar macro is a slice that provides four separate 6-input look-up tables. With one LUT, the multiplexer for vertical routing in front of a particular I/O bar can be implemented directly within the I/O bar macro itself (see also Figure 5). Likewise, the two alignment multiplexers for connecting a module to a signal path of an I/O bar can be implemented in one single Virtex-5 LUT. This is possible as one 6-input LUT can be split into two separate 5-input LUTs that have to share the same inputs. Therefore, two LUTs are required to implement the I/O bar per signal wire within the static system (one for the start connection and one for the endpoint). In the case a module is connected to the bar, one LUT will be required per used I/O bar signal. For a module connection, it is possible to configure to use the flip-flop (that is attached to the LUT) for either the downstream, or the upstream, or neither of them.

Figure 6. Maximal achievable clock frequency over the width of an I/O bar in terms of CLBs for a Xilinx Virtex-II Pro and a Virtex-5 FPGA. The gaps in the curves result from special resource columns like block RAMs. The figure lists the routing delay while omitting FF setup&hold times.

Because of the LUT packing into slices, n I/O bar signals cost $L_s = 2 \cdot 4 \cdot \lceil \frac{n}{4} \rceil$ LUTs for the static connection and $L_m = 4 \cdot \lceil \frac{n}{4} \rceil$ LUTs for a module connection. As a consequence, I/O bar signals should be grouped in wiring bundles of 4 wires each. Figure 2 points out that 10 (11) wires can be used in one CLB to implement an I/O bar towards east (west) direction. However, only 6 routing wires start in each CLB that would allow to implement a regular structured communication architecture without additional hops. Consequently, implementing one 4 wire bundle per CLB row is optimal. Furthermore, this will leave sufficient routing resources for implementing the reconfigurable modules. The impact of fitting modules into bounding boxes and of allocating wire resources for a communication architecture was examined in [4].

C. I/O Bar Throughput

The most important performance measurement of a communication architecture is the throughput that can be achieved between the static system to and from the reconfigurable modules as well as among the modules. When considering a fully pipelined I/O bar with flip-flops at all inputs and outputs, the critical path occurs if no module is connected to the I/O bar. This is because a connected module would act as a pipeline stage on the I/O bar routing path. The achievable clock frequency (and consequently the throughput) over the I/O bar width is revealed in Figure 6. Surprisingly, the achievable clock frequency of the I/O bar that is implemented on the 130 nm Virtex-II is almost identical to the bar that has been designed for the 65 nm Virtex-5 device. The reason for this is that cascading two wires that each route only one CLB further introduces an additional latency as compared to the double lines (wires that span a distance of two CLBs) that are used for the Virtex-2 I/O bar implementations.

The throughput is the product of the clock frequency and the number or wires that are used for implementing the I/O bar. When considering the afore mentioned bound of 4 signals per row of CLBs, a resource slot would provide up to $20 \cdot 4 = 80$ individual wires towards one direction within the height of a configuration frame of a Virtex-5 device. For a system providing a reconfigurable area that is 30 CLB columns wide, this would result in a maximum throughput of 2 GB/s towards one direction, when running the I/O bar at 200 MHz.

With the help techniques like pipelining and multipath routing, it is possible to enhance the size of an I/O bar beyond the limit that is listed in Figure 6. For instance,

Figure 7. Physical implementation of a static system consisting of $4 \times 30 = 120$ resource slots for hosting reconfigurable modules. The system provides four sets of I/O bars to integrate reconfigurable modules at runtime. The vertical routing is implemented as illustrated in Figure 5. a) Defining prohibit regions for the reconfigurable region works stable for the placement of logic resources, but not for the routing when using the standard Xilinx tools. b) By adding blocker macros that congest all routing resources that are not part of the communication architecture, the router will be forced not to route through the static region, as depicted in c).

using two bars in parallel at half speed (two cycle path) would allow to implement an I/O bar that spans over the full width of a Virtex-5 5VLX110t-3 device, while running at 200 MHz. Such an I/O bar would require the double amount of wire resources and consequently limiting the accumulated throughput to 1 GB/s. As an alternative, connection points for just pipelining the data on the I/O bar wires might be set regularly within the static part and/or the modules. For example, if the system is designed in such a way that an I/O bar signal is passed through a flip-flop after a routing distance of less than 10 CLBs, a clock frequency of up to 500 MHz can be achieved.

IV. DESIGN FLOW

For the physical implementation, a tool called *ReCoBus-Builder* has been implemented that is available on www.recobus.de. This tool generates the communication architecture as one or more I/O bar hard macros. As far as possible, ReCoBus-Builder tries to hide FPGA low level details from the designer. Besides the communication architecture, ReCoBus-Builder assists in the floorplanning and the generation of constraints. In this step, one or more partially reconfigurable areas will be defined that will be prohibited for implementing any static logic or routing. In addition, bounding boxes will be generated for encapsulating reconfigurable modules in one or more resource slots, that are located within the reconfigurable areas. Note that slots may be cascaded in any horizontal or vertical manner. This might enormously enhance the resource utilization. For example, a module requiring much block RAM can be narrow but height to access the RAM, while a pure logic only module may scale more towards a horizontal extend. The static system, as well as all partially reconfigurable modules are implemented independently from each other.

The here presented tool flow is different to the flow Xilinx provides around their floorplanning tool PlanAhead [5]. While only one module can be placed exclusively in a reconfigurable area using the PlanAhead flow, our tools allow to tile a large reconfigurable area in multiple tiny resource slots that can be assigned very flexible to multiple

modules at the same time and in a two-dimensional fashion. This allows it, for example, to replace one large module with multiple smaller ones. A further big difference, which makes the PlanAhead flow difficult to handle, is that modules are closely related to the implementation of the static system, as local routing resources within the reconfigurable region might be used for implementing static routing. Consequently, the reconfigurable modules will have to include the configuration of these local routing resources. This prevents to relocate modules to different positions on the FPGA.

The Xilinx ISE design suite does not provide stable working routing constraints. As can be identified in Figure 7a), this affects the reconfigurable area. However, by generating special blocker-macros we can force the router to prohibit reconfigurable regions from being used to implement any static routing.

V. CONCLUSION

While newer FPGA families offer advantages in logic density, power consumption, and speed, the implementation of runtime reconfigurable systems has become more difficult. This paper discussed trends in newer FPGA architectures with respect to an implementation of runtime reconfigurable systems. With a trend towards more logic and less routing resources (that are also more irregular arranged as in previous architectures), implementing an on-FPGA communication architecture has become a challenge.

We extended the ReCoBus-Builder tool with a special adjusted communication architecture that can be scaled to a throughput of several GB/s. This is possible while being capable of integrating modules in a very fine-grained two-dimensional tile grid, thus, significantly reducing the internal fragmentation when fitting modules into tiles. With this high performance and efficiency improvement, combined with an easy usable design flow, we supplied the technical basis for implementing sophisticated runtime reconfigurable systems on newer FPGAs, such as Virtex-5.

ACKNOWLEDGMENT

This work is supported by the Norwegian Research Council under grant 191156V30

REFERENCES

- [1] D. Koch, et al., "Minimizing Internal Fragmentation by Fine-grained Two-dimensional Module Placement for Runtime Reconfigurable Systems," in *17th Symposium on Field-Programmable Custom Computing Machines (FCCM)*. Napa, CA., USA, Apr. 2009, pp. 251–254.
- [2] S. Douglass, et al., *The Next Generation 65-nm FPGA*, may 2006, talk at the Hot Chips Symposium (HC 18), source: <http://www.hotchips.org/archives/hc18/>
- [3] T. Becker, et al., "Enhancing Relocatability of Partial Bitstreams for Run-Time Reconfiguration," in *Proceedings of the 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*. Napa, Ca., USA: IEEE, 2007, pp. 35–44.
- [4] D. Koch, et al., "A Communication Architecture for Complex Runtime Reconfigurable Systems and its Implementation on Spartan-3 FPGAs," in *Proceedings of the 17th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*. Monterey, Ca., USA: ACM, Feb. 2009, pp. 233–236.
- [5] N. Dorairaj, et al., *PlanAhead Software as a Platform for Partial Reconfiguration*, 2005, 4th quarter, Xilinx XCELL journal, available online: http://www.xilinx.com/publications/xcell-online/xcell_55/xcell_pdf/xcell_pmethod55.pdf.