DOI: 10.35598/mcfpga.2019.004

Field Programmable Counter Arrays Integration with Field Programmable Gates Arrays

Vladimir Karnaushenko
Dept. of Microelectronics, electronic devices and appliances
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
vladimir.karnaushenko@nure.ua

Alexander Borodin
Dept. of Microelectronics, electronic devices and appliances
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
alexander.borodin@nure.ua

Abstract–Field Programmable Counter Arrays (FPCAs) have been recently introduced to close the gap between Field Programmable Gates Arrays (FPGA) and Application Specified Integrated Circuits (ASICs) for arithmetic dominated applications. FPCAs are reconfigurable lattices that can be embedded into FPGAs to efficiently compute the result of multi-operand additions.

Keywords–Field Programmable Counter Arrays, arithmetic applications, integration, shadow cluster.

I. FPCA INTEGRATION WITH FPGAS

This thesis presents a study of the issues related to integration of hard blocks (and/or coarse-grained blocks) into FPGAs. It then proposes some integration scenarios for FPCAs and describes a generic platform for implementation and evaluation of some of these scenarios based on Stratix II devices and the FPCA architecture.

The Problem

The introduction of hard logic blocks and coarse-grained blocks for FPGAs creates a new problem: their seamless integration. In simple words, the problem asks how should these blocks be floor planned and placed in the homogeneous array of soft logic, and how should they be connected to the routing fabric efficiently? The floor plan should result in shorter critical paths and reduced congestion and an interface must be designed for the block that meets the following requirements:

- it should provide the required level of connectivity (i.e. all typical circuits using the block should be routable);
- it should be fast and consume minimum chip area;
- it should minimize the negative impact on the rout ability of other blocks.

II. RELATED WORKS

Field Programmable Counter Arrays (FPCAs) are one-dimensional array of basic computational elements called Compressor Slices (CSlices). FPCAs are configurable lattices that perform Multi-Operand Additions (MOA) efficiently. MOAs – either explicitly or implicitly in the heart of other blocks – occur frequently in arithmetic circuits used in video applications, cryptography, wireless communication, etc. In multipliers, the partial product bits generated by a level of AND gates, represent MOA as well.

Dadda and Wallace trees reduce the partial products to a two input addition. They are also referred to reduction trees. Verma and Ienne have proposed a set of transformations which expose large multi-operand additions from arithmetic circuits. In this way, datapath circuits can be be implemented more effectively by specific digital circuits like FPCAs (also called here compressor trees) rather than general logic produced by using commercial synthesis tools.

Although there has been significant study on new architectures for hard and coarse-grained blocks for FPGAs, few of them have studied their detailed interface. In [1], formal optimization methods are used to design mixed-granularity FPGA architectures. Integer Linear Programming (ILP) is incorporated to determine the best floor plan to optimize the architecture for a set of DSP applications, including the choice of the best mix of hard 18*18-bit multipliers.

A similar problem is studied for block RAMs in [2]. In this work, without any investigation and inspired by commercial FPGAs, it is assumed that a row of block RAMs is located in the middle of the chip (like Figure 1). The authors have tried to determine the ideal flexibility of the memory/logic interconnect block (illustrated in Figure 2). The flexibility of a memory/logic block is defined as the number of (or portion of) available routing wires to which each memory pin is connected. This study shows that if the flexibility is too low, many circuits become unroutable, while excessive large flexibility values increase the memory access time and also waste chip area.

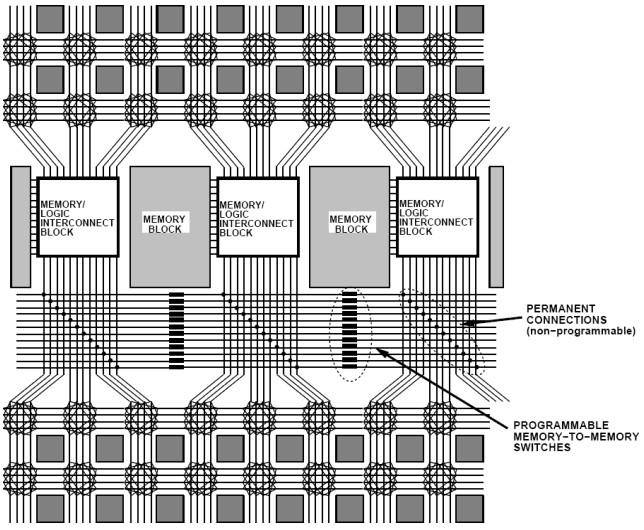

Fig. 1. An example of integrating RAMs as hard blocks [1].

Alternatively, the authors have made several enhancements to the routing architecture based on the characteristics of memory-to-memory connections, such as busses, in their benchmark circuits. Since nets connecting to multiple memory blocks are common in many circuits' blocks, the authors have proposed to add additional programmable switches between adjacent memories to support these nets. This significantly improved the results on architectures with lower interconnect block flexibility.

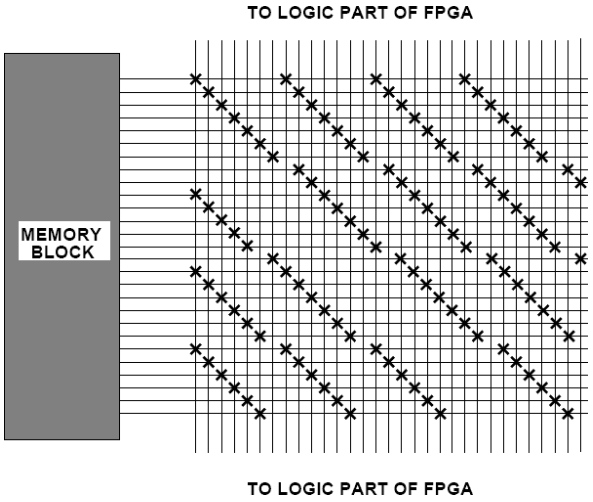

Fig. 2. Example of memory/logic interconnect block [1].

The large M-RAM blocks in Stratix II device resemble this style of integration. This solution enhances the ability to tile island-style architecture, and requires a completely new design for interfacing with the rest of routing fabric. Greater integrity and speed are achieved with larger hardwired blocks, but the layout design and interface design becomes a more complicated.

It doesn't seem that the results obtained for memory block integration could be used for arithmetic blocks such as FPCAs. The functionality of the pins and their contribution to total routing resource demand are different for blocks with different functionalities.

A very recent work [3], has studied the integration of coarse grained Floating Point Units (FPUs) in a fine-grained soft logic array. Different floor planning strategies for placement of the FPUs, different aspect ratios and possible pin placement methods are evaluated to find the optimum architecture. The approach taken is again an empirical one based on the delay and minimum channel width requirement of a set of benchmarks. Unlike the previous approach, they have assumed that the gridded routing fabric extends over their Embedded Blocks (EBs). Figure 3 shows a scenario where a 3*3 super-tile is replaced by an embedded block.

The M512 RAMs, M4K RAMs, and the DSP blocks in Stratix II devices are examples of this approach, but with a small difference. Tiles in the same column are all of the same kind. These tiles are all the same height (or multiples of same height) but their widths may slightly differ. In this way, the general routing fabric could be designed as easily as the general island-style routing fabric consisting of horizontal and vertical channels of routing wires with switch blocks in their intersections points. The problem of interconnect interface block design in this approach; will be to minimize the re-design of the intra-cluster connections in such a way that matches the actual pin-demand of the new hard blocks.

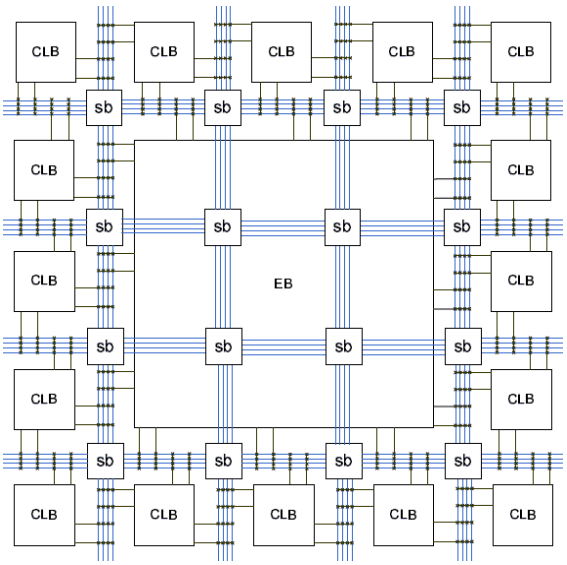

Fig. 3. Expansion of the gridded routing fabric over the embedded block [3].

As an example, the DSP blocks in the Stratix II architecture span 4 blocks vertically. The blocks are designed in such a way that they can be decomposed into four tiles. Each tile has the same height as other logic tiles and has a switch box, intra-cluster connections and the DSP core itself. The intra-cluster connection design for DSP blocks is interesting. LAB tiles in Stratix II devices have 45 local interconnect lines that are selected by a level of switches from the general routing network. These lines drive all the ALM inputs which are around 65 input pins. For the DSP tiles (¼ each DSP block), there are 60 local lines that drive approximately 40 input pins. This information is summarized in table 1. The reason for this local interconnect-input pin difference is that it is the actual pin-demand of the tiles which is important, not just the number of input pins. Many of the 65 input pins of the ALMs in each LAB could be shared or driven by the local feedback lines. This lowers the actual pin demand to 44. On the other hand, DSP block input pins are arithmetic bits, which are all distinct, and needed to be routed separately. Thus, more connections than the total number of input pins are provided by the local lines to ensure the required routing flexibility. FPCAs, from this point of view, are more similar to DSP blocks than to block RAMs.

TABLE I. INTRA-CLUSTER DESIGN OF LAB AND DSP TILES

Tile Type	*Local Interconnect Lines*	*Input Pins*
LAB Tile	44	=65
DSP Tile	60	=40

Hard blocks improve the area and speed of the designs mapped to FPGAs, but only if they are used. Otherwise, the silicon area devoted to them and, the expensive routing resources around them are wasted. This also suggests that the integration of hard blocks is only feasible if they are used often. Shadow clusters are introduced in [4,5] to take

better advantage of the routing resources around hard blocks, when they are not used. A shadow cluster is a soft logic block, placed "behind" the hard block so that if the design doesn't use the hard block, then some general FPGA logic within the shadow cluster can be used to implement a portion of the real circuit. Shadow clusters come at the expense of additional area, but, if properly used, the advantage obtained by making better usage of the routing network dominates this extra area overhead. Figure 4 depicts this idea. The inputs, which come from the routing network, are shared between the shadow cluster and the hard block. Depending on the mode of operation, either the output of hard block or the shadow cluster is selected.

Design Space Exploration (DSE) is a method to tackle problems where an analytical approach is difficult to take or there is no analytical solution based on the available theories and models. FPCA architecture design – according to our investigations – falls into this group of problems. By twisting every single knob in FPCA architecture, two trends affecting the performance in opposing directions could be identified that suggest the existence of an optimum point for each parameter. Alternatively, this optimum point depends on the value of other parameters, the technology used for VLSI implementation and, most importantly, the application (benchmarks) being mapped on the FPCA.

For example, increasing the MORC of the CSlices reduces the number of CSlices required to synthesize an application on the FPCA, improves the performance by making the critical path pass through fewer output multiplexers, and saves area by using fewer first-level counters. But, if the configuration of the GPCCC or the characteristics of the benchmarks does not allow exploitation of output ranks, thicker output multiplexing layers decrease the performance, and the area dedicated to extra parallel counters columns in the CSlices are wasted. An empirical approach could help overcoming such problems by examining all possible points in the design space, which can not be identified just by analysis. Since the intention is to have a real hardware model on which the benchmarks could be mapped and the area/delay values be extracted, the model is developed using synthesizable subset of VHDL.

Each FPCA sub-block was modeled in a generic fashion and sub-blocks were connected together in higher level blocks (also generic). In VHDL, generic statements are used to model generic blocks. Some of these generic values are calculated using a Perl script and written to a VHDL package which is included by other modules. The rest of the model is developed in pure VHDL.

Developing a generic HDL model of FPCAs was a non-trivial task.

Two of the most significant challenges were (1) Modeling parallel counters in an efficient way and (2) Modeling the interconnection of components inside a CSlice.

The first approach taken for modeling parallel counters was using behavioral VHDL as a loop in a process statement which counts the input bits and produces outputs. These models were synthesized using Synopsys Design Compiler v2006.06 and the compile_ultra optimization capability of the tool. The result for a 31:5 counter was poor. The synthesis tool could not find an efficient way to restructure the counter to produce acceptable results. One of the well known ways for efficient implementation of parallel counters is using a tree of Full-Adders and Half-Adders [6]. In this work, based on the ability of VHDL to model recursive circuits [7] a generic adder tree is modeled to mimic a tree of full adders and half adders. The results obtained by this approach were more acceptable and comparable to manual description of fixed size counters. More advanced methods for synthesis of parallel counters are also suggested [8].

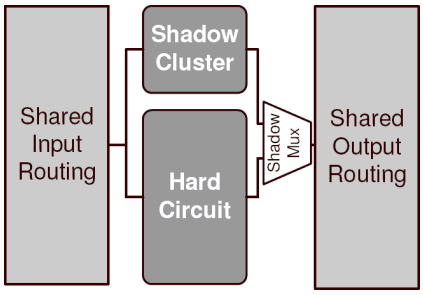

Fig. 4. Illustration of shadow cluster concept [3, 4].

III. Conclusions

A design space exploration tool for FPCAs consisting of a generic model of FPCAs, a mapping heuristic with synthesis and report automation facilities were developed. An analysis of the design space was performed and a new metric called utilization was suggested to prune the DSE.

A set of benchmarks were chosen and the DSE were performed, and some of the best performing architectures in terms of speed and area were highlighted.

The problem of integrating FPCAs with FPGAs was also studied.

References

[1] A.M. Smith, G.A. Constantinides, and P.Y.K. Cheung, "Integrated Floorplanning, Module-Selection, and Architecture Generation for Reconfigurable Devices," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 16, 2008, pp. 733-744.

[2] S. Wilton, J. Rose, and Z. Vranesic, "The memory/logic interface in FPGAs with large embedded memory arrays," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 7, 1999, pp. 80-91.

[3] C.W. Yu et al., "The Coarse-Grained / Fine-Grained Logic Interface in FPGAs with Embedded Floating-Point Arithmetic Units," Programmable Logic, 2008 4th Southern Conference on, 2008, pp. 63-68.

[4] Peter Jamieson and Jonathan Rose, "Enhancing the area-efficiency of FPGAs with hard circuits using shadow clusters," Field Programmable Technology, 2006. FPT 2006. IEEE International Conference on, 2006, pp. 1-8.

[5] P. Jamieson and J. Rose, "Architecting Hard Crossbars on FPGAs and Increasing their Area Efficiency with Shadow Clusters," Field-Programmable Technology, 2007. ICFPT 2007. International Conference on, 2007, pp. 57-64.

[6] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford University Press, USA, 1999.

[7] P. Ashenden, "A comparison of recursive and repetitive models of recursive hardware structures, *VHDL International Users Forum. Spring Conference, 1994. Proceedings of*, 1994, pp. 80-89.

[8] A. Verma and P. Ienne, "Automatic Synthesis of Compressor Trees: Reevaluating Large Counters," *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07*, 2007, pp. 1-6.