An FPGA Configuration Stream Architecture Supporting Seamless Hardware Accelerator Migration

Christophe Huriaux, Olivier Sentieys, and Antoine Courtay
INRIA/IRISA – University of Rennes 1
christophe.huriaux@irisa.fr, olivier.sentieys@irisa.fr, antoine.courtay@irisa.fr

***Abstract*—This paper introduces a novel approach to hardware task relocation in an FPGA-based reconfigurable fabric, allowing offline design, routing, and unfinalized placement of hardware IPs and dynamic placement of the corresponding bitstreams at run-time. Our proposal relies on a custom dual-context FPGA configuration memory organization in a shift-register manner and on a dedicated bit-stream insertion controller leading to a break-through in terms of adaptative capabilites of the reconfigurable hardware. We show that using our custom shift-register organization across the configuration memory, and under some weak constraints, we can greatly reduce the overhead implied by the 1-D to 2-D mapping of the shift-register onto the logic fabric.**

INTRODUCTION

Field Programmable Gate Arrays (FPGA) provide the flexibilty, performance and power efficiency required by modern applications in the growing market of integrated circuits, in contrast with dedicated processors or Application-Specific Integrated Circuits (ASIC) which require higher budget and developement time.

Most of the available commercial FPGAs use an addressable memory organized around an array of N-bits words of Static RAM (SRAM) cells. Such configuration memory is traditionally programmed by writing, to each word, the corresponding bitstream data at runtime.

In the developing domain of Dynamically Partial Reconfiguration (DPR), parts of the hardware can also be self-reconfigured while the rest of the chip is still running. Dynamically reconfigurable hardware tends to add another layer of flexibility, at the cost of more complex design stages. This technique enables the fabric to be adaptive to various computational tasks and to modifications of application context and environment through hardware fabric reuse.

SRAM-based configuration memories will be relocated in two-steps in the context of DPR [1]: each configuration word of the relocatable area will be read from its original location and then written to the new one, with a shutdown period of the two areas during the relocation in order to prevent undesirable side-effects.

These specific aspects of the SRAM memory array for DPR introduce a non negligible overhead for time-multiplexing applications where hardware IPs needs to be frequently loaded and/or relocated on the logic fabric.

In this paper, we propose a new organization of the FPGA configuration memory in a shift register manner (hereafter named the scan-path, due to its similarities with the verification world) which allows the logic fabric to ease task relocation inside the configuration memory. At the cost of a small area overhead, our memory organization proposal is able to perform near-instant task switching thanks to a double layer memory. The work presented in this paper mainly focuses on the hardware design of the configuration memory.

This paper is organized as follows. In Section I, we briefly discuss some of the dynamically reconfigurable hardware architectures available solutions. In Section II we introduce our proposed architecture. Some experimental results considering different memory organizations are also presented. Finally, in Section III, we give some insights on possible future work.

I. RELATED WORKS

The only commercially available fine-grained dynamically reconfigurable architectures are found in FPGAs [2]. The Xilinx Virtex-7 latest architecture allows to design a partially reconfigurable system where slices could be reconfigured at runtime under certain conditions. Reconfigurable locations have to be defined during the design stage, which leads to long bitstreams, dedicated to their final location in the fabric.

Different approaches have been proposed in the literature to counterbalance the limitations of Xilinx FPGA [3] and offer online task relocation to existing circuits, with important reconfiguration costs. Research topics also include more specifically task relocation and defragmentation on Xilinx XC6200 partially reconfigurable FPGA [4] [1] which is, however, less than ideal for relocation since the programming architecture of the circuit is not designed for that purpose.

In contrast, other research tracks have been conducted into the field of multicontext FPGA [5] [6] where each configuration bit comprises in fact multiple memory cells. This technology allows to independently program each of the memory layers and then to switch between them. This method has however a great impact on the overall area used by the configuration memory.

Our approach aims at reducing the time penalty induced by dynamic reconfiguration and task migration on a reconfigurable hardware. Thanks to a novel design of the FPGA configuration memory and dedicated controllers, a given Intellectual Property (IP) bitstream can be reused for multiple insertions and locations.

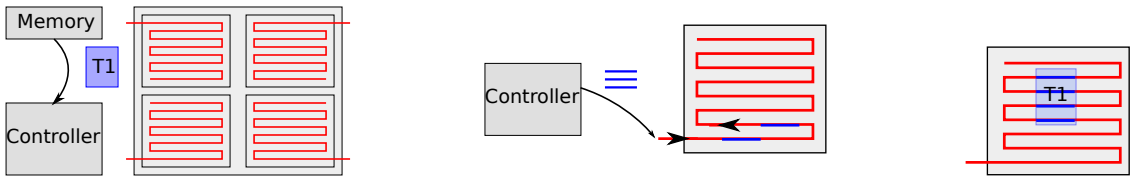

Fig. 1. The three steps for task insertion in our proposed architecture: (a) IP download from memory (b) Decompression and bit-stream serialization (c) Final placement

II. Current work

A. Hardware architecture

The proposed architecture is designed to be partially reconfigurable: only a portion of the chip can be reconfigured while the remaining logic elements can pursue their operations normally. This behavior is achieved thanks to a virtual split-up of the whole logic fabric into several *domains* whose sizes are fixed during the design stage of the architecture, as shown on Fig. 1a.

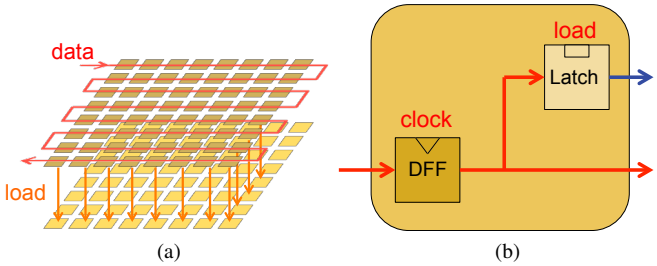

Fig. 2. (a) Overview of the double-layer memory. (b) A double-context cell.

Each of these domains contains an independant configuration memory which holds the state of the logic and routing elements. While traditional memories of this kind are mostly based on a SRAM grid with word-adressing for read and write operations, our architecture introduces a set of configuration bits organized as a shift-register where each of the D Flip-Flop (DFF) of the register is doubled with an SRAM cell materializing the real configuration bit as depicted by Fig. 2b. Once the shift-register is fully populated with the task content, the configuration held by the DFF is loaded in a single clock cycle into the SRAM cell of the second memory layer retaining the real configuration bits, as shown on Fig. 2a. This mechanism helps to reduce the reconfiguration costs in terms of configuration domain downtime, but it increases the area footprint of each double memory cell.

This double context memory is thus able to receive a bit-stream without requiring an outage of the functional memory array, which allows to load an IP configuration wihtin a cycle once the shift-register has been loaded. The main consequence of these memories is the implied area overhead. We ran simulations and synthesis of different custom double layer memories with up to 16K cells using a 65nm technology and extracted area information from the results. Table I shows the extra area needed compared to a single layer SRAM memory typically used in FPGA, and compared to hybrid memory described in a Xilinx patent [7] which eliminates the need for a regular array of address lines in the SRAM memory.

Memory organization	Area overhead	
	w.r.t single SRAM	w.r.t. Xilinx hybrid
Double SRAM	+101,8%	+38,5%
Double layer scan-path	+74,8%	+20%

TABLE I. Area overhead of a double-layer configuration memory.

The double SRAM is the basic implementation of a dual layer memory: a first layer is word-addressed and can be loaded to a second layer, thus the area needed is substantially twice as large as the single layer version. Due to the lack of addressing lines in our organization proposal the area overhead of the DFF shift-register is counter-balanced, which leads to a 25% smaller area overhead in comparison to a double SRAM memory. This memory organization offers easier layout for the same reason, it also has stronger optimization possibilities in terms of energy comsumption for future work.

The domains of the logic fabric have thus a single entry-point through which the entire bit-stream of an IP will be input. This IP bit-stream is stored in memory in an unfinalized and compressed form which allows to place the hardware task at any given location of the logic fabric without the need of online routing. A dedicated piece of hardware is responsible for on-the-fly *devirtualization* of the virtual bit-stream into a placeable IP bit-stream, which is inserted into the shift-register with additional padding in order to perform a good alignment of the task on the logic fabric, as depicted by Fig. 1b and 1c.

B. Configuration stream organization

The overall efficiency of the architecture in terms of relocation possibilities strongly depends on the shift-register organization among the configuration memory cells. Since the goal is to insert a 1-D bit-stream (bit sequence) into a 2-D memory matching the logic element position. Our proposal takes part of existing fractal designs known as space-filling curves which realize a mapping from a curve in the unit interval $[0;1[$ into the unit square $[0;1[\times[0;1[$.

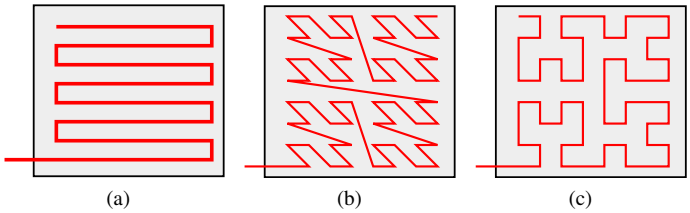

Fig. 3. Examples of shift-register dispositions : (a) Naive (snake), (b) Z-order, (c) Hilbert

Space-filling curves have already been heavily studied for their properties in other research fields. Our proposed architecture will use the geometrical capabilities in order to build the shift register (based on such a fractal construction scheme). These fractal organizations allow a more scalable way to deal with the insertion of IPs in the shift register. As an example, Z-order and Hilbert fractal curves are by nature composed of a single shape repeated at different scales as seen in Fig. 3b and Fig. 3c. They will eventually lead to more relocation possibilities by only shifting the register to move the task into its domain.

C. Task relocation

As stated in the previous section, it is possible to take part of especially crafted scan-path organizations to enhance the relocation possibilities of a given domain. We can show that, using the naive *snake-like* shift-register disposition (depicted in Fig. 3a) among homogeneous configuration cells and considering a task with dimension $w \times h$ logic cells which has been originally placed at position (x, y) in a logic fabric of $W \times H$ logic cells, the migration possibilities of this task will be limited to a set Δ of destination coordinates, where

$$\Delta = (x, y + 2k) \qquad \forall k \in \left[0, \frac{\lfloor H - h - y \rfloor}{2}\right].$$

It means that the task can only be migrated in the same column without requiring either a costly mirror and/or rotation transform or a complete regeneration of the shift-register. The later operation is also very costly because it forces all the running logic of a given domain to be shut-down, and then to reload each of the running IPs from the memory. A new *domain bit-stream* needs therefore to be built, mixing the logic cells configuration bits of each running hardware task at the good position before the insertion of this data into the scan-path.

A tradeoff has to be found between the domain size and the logic fabric versatility in order to balance

- the relocation possibilities of a task,
- the cost of a full shift-register rebuild when a shift-only task migration could not be determined.

D. Scan-path complexity considerations

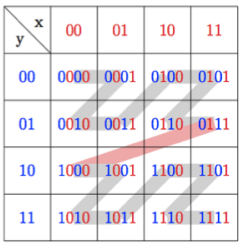

Fig. 4. Mapping from 2-D binary coordinates to a 1-D offset

Each of the studied fractal organization has its own pros and cons, e.g. the Z-order curve has the strong advantage to only require wired logic to map a 1-D offset of the bit-stream to the 2-D coordinates in the logic fabric, as shown on Fig. 4. For a given 2-D coordinate $(x; y) = [(\mathbf{2}; 3)]_{10} = [(\mathbf{10}; 11)]_2$ the corresponding 1-D offset in the shift-register is trivially determined by interleaving the x and y components bits offset $= [1\mathbf{11}\mathbf{0}]_2 = [14]_{10}$.

It is thus a good candidate for a fast on-the-fly shift-register content elaboration. The drawback of this curve is its composition, by nature, of very long lines which cross an entire domain and therefore increase the critical-path of the shift-register (a consequence which is balanced by the size of the domain).

On the other hand, like all of the Peano space-filling curves, the Hilbert curve has a fairly complex mapping from 1-D to 2-D relying on a set of recursive equations defined by

$$x_h(0) = 0, \quad x_h(1) = 1, \begin{cases} x_h(\frac{t}{4}) & = \frac{y_h(t)}{2} \\ x_h(\frac{t+1}{4}) & = \frac{x_h(t)}{2} \\ x_h(\frac{t+2}{4}) & = \frac{1+x_h(t)}{2} \\ x_h(\frac{t+3}{4}) & = \frac{2-y_h(t)}{2} \end{cases} \tag{1}$$

and

$$y_h(0) = 0, \quad y_h(1) = 0, \begin{cases} y_h(\frac{t}{4}) & = \frac{x(t)}{2} \\ y_h(\frac{t+1}{4}) & = \frac{1+y(t)}{2} \\ y_h(\frac{t+2}{4}) & = \frac{1+y(t)}{2} \\ y_h(\frac{t+3}{4}) & = \frac{1-x(t)}{2} \end{cases} \tag{2}$$

.

Although optimizations could be reused [8], building a hardware implementation is more subject to resource waste. However, the cells have a better distribution in the plan in comparison to the Z-order curve. This organization also offers rotations of a given bit-stream with only shifts of the configuration memory, which leads to more relocation possibilities if the logic fabric is fairly regular.

E. Performance analysis

We analyzed the performance of the three studied shift-register organizations in terms of

- number of operations (*i.e.* the real number of shifts needed to place an IP),
- relocations possibilities without regenerating the bit-stream, considering either straight placement or including rotated tasks.

Since we are targeting a shift-register based architecture (*i.e.* with a single input and a single output), we cannot arbitrarily place a given task data at a specified $(x; y)$ origin without adding padding bits whose sole purpose is to align the 1-D data inserted in the 2-D fabric. In the case of a single task placed on the fabric, these padding bits are useless and add unnecessary cycles to the task placement. If two tasks are placed close together, the reconfiguration controller may need to fetch data from the two tasks when inserting each one on the fabric, ultimately leading to an increased placement time.

Therefore, our results take this overhead into account: it has to be interpreted as the number of additional bits required

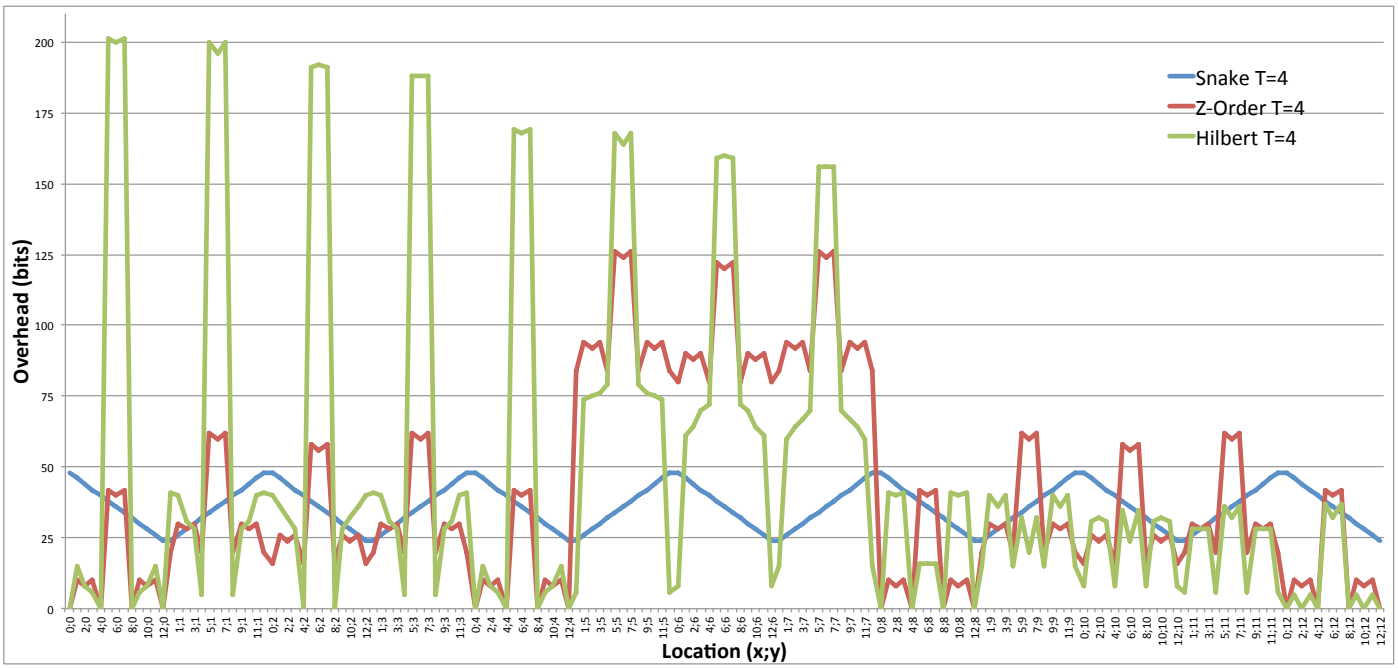

Fig. 5. Overhead of the insertion of a 4×4 task in a 16×16 fabric

to insert a task at a specific condition. Figure 5 summarizes this overhead in the case of a 4×4 task inserted at all possible positions of a 16×16 fabric. This graph depicts the performances of the three scan-path organizations and raises the weaknesses of each one.

The naive snake organization offers poor performances as its overhead is always non-zero and oscillate in a sawtooth manner as the task moves from a line to another. Hilbert and z-order curves have a much complex behavior, though. Even if local maxima of the graph implies a much greater overhead than the snake organization, these curves offer nonetheless a lot more zero or near-to-zero overhead location possibilities. Additionally, the fractal curves have a stronger scalability, whereas the minimum overhead implied by the snake organization is linearly dependent of the fabric size for a given task size.

In the following results, we have done the assumption that weak constraints could be defined in order to ensure good performances: particularly, we assume that the granularity of the task location may be decreased in order to promote lower overheads. In the case of Hilbert and z-order curves, a good constraint is to limit the task to an even size only, and to place the tasks onto boundaries which are a multiple of the task size (*e.g.* a 4×4 task should be placed at x positions $0, 4, 8, 12$ and so on). Although not mandatory this constraint greatly reduces the overhead as stated earlier, as the boundary comes closer to the task size (up to zero overhead when the boundary and task size match exactly as shown on Table II).

Organization	Mean overhead		Relocation	
	Even pos.	Task mult.	Straight	w/ rotation
Snake	+225%	+225%	4	7
Z-order	+150%	0%	16	16
Hilbert	+180%	0%	6	16

TABLE II. PERFORMANCE ANALYSIS WITH 4×4 TASKS IN A 16×16 FABRIC

The fabric size versus task size ratio have a strong impact in the relocation performance: In order to ensure better performances relatively to a naive scan-path organization, the fabric size have to be at least thrice as large as the maximum task size, as depicted by Tables II and III

Organization	Relocation	
	Straight	w/ rotation
Snake	3	5
Z-order	4	4
Hilbert	1	3

TABLE III. PERFORMANCE ANALYSIS WITH 8×8 TASKS IN A 16×16 FABRIC

The improved locality of 2-D coordinates and 1-D offsets of space-filling curves is a good property to reduce the insertion overhead. In order to compare them, we define t_1 and t_2 two bitstream offsets, $X_1 = (x_1; y_1)$ and $X_2 = (x_2; y_2)$ their corresponding 2-D coordinates on the fabric given a specific shift register organization, and $\vec{T_{12}}$ and $\vec{X_{12}}$ the respective 1-D and 2-D vectors of the previously defined component.

In the case of the naive scan-path, the distance $\|\vec{T_{12}}\|$ is not necessarily close to $\|\vec{X_{12}}\|$, and this is a frequent case at the border of the fabric. Considering the Z-order curve we can show that this locality is greatly improved, but there is still a problem in the "Z" breaked line, and it gets worse when the fabric is scaled up. The Hilbert curve does not have this locality problem, for every given $\|\vec{X_{12}}\|$ the corresponding $\|\vec{T_{12}}\|$ will have the same order of magnitude. Considering this property, the resulting "ready to insert" bitstream will have a higher compactness: there will be little to no need to insert padding bits in order to match the 2-D geometry of the shift-register.

III. CONCLUSION

In this paper we have proposed a novel configuration memory architecture which uses a double context shift-register

to hold the configuration data in order to balance the area overhead of the multi-context memory and the improved flexibility. An actual performance gain is added with a custom design of the shift-register organization on the 2-D plan which rely on an application of space-filling curves. Those curves have interesting properties when applied to the design of our configuration stream memory, especially when we look at the the overhead of padding bits which needs to be added into the stream. When inserting a given IP, the locality induced by the Hilbert curve helps to greatly reduce the number of those bits (and thus of computation needed) in most cases.

We have shown that under some weak constraints on the task placement, the bit padding overhead required to match the 2-D geometry of the fabric could be greatly reduced while using our proposed shift-register architecture, leading to better results than the naive snake-like organization. The fractal organizations also leads to a better scalability which is a strong advantage when targetting real hardware implementation.

Our further research will focus on the integration of relocatable tasks within a heterogenous context by taking advantage of the fractal shift-register organization and defining variably shaped domains at runtime in order to balance the lower relocation possibilities introduced by heterogeneous resources.

ACKNOWLEDGMENT

This research is sponsored by the European Commission under the 7th Framework program within the FlexTiles project (FPT ICT-288248) and by the French Ministry of Research.

REFERENCES

[1] K. Compton, Z. Li, J. Cooley, S. Knol, and S. Hauck, "Configuration relocation and defragmentation for run-time reconfigurable computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 10, no. 3, pp. 209 –220, June 2002.

[2] K. Compton and S. Hauck, "Reconfigurable computing: a survey of systems and software," *ACM Computer Survey*, vol. 34, no. 2, pp. 171–210, June 2002.

[3] H. Kalte and M. Porrmann, "Replica2pro: task relocation by bitstream manipulation in virtex-ii/pro fpgas," in *the Proceedings of the 3rd conference on Computing frontiers*, ser. CF '06. ACM, 2006, pp. 403–412.

[4] *XC6200: Advance Product Specification*, Xilinx, Inc., 1996.

[5] A. DeHon, "DPGA utilization and application," in *the Proceedings of the 1996 ACM fourth international symposium on Field-programmable gate arrays*. ACM, 1996, pp. 115–121.

[6] S. Trimberger, D. Carberry, A. Johnson, and J. Wong, "A time-multiplexed fpga," in *the Proceedings of the 5th Annual IEEE Symposium on FPGAs for Custom Computing Machines*. IEEE, 1997, pp. 22–28.

[7] P. Rau, A. Ghia, and S. Menon, "Configuration memory architecture for FPGA," April 2001, US Patent 6,222,757.

[8] A. Butz, "Alternative algorithm for Hilbert's space-filling curve," *IEEE Transactions on Computers*, vol. 100, no. 4, pp. 424–426, 1971.