# DYNAMIC DATA-PATH SELF-RECONFIGURATION OF A VLIW-SIMD SOFT-PROCESSOR ARCHITECTURE

*Guillermo Payá-Vayá, Roman Burg, and Holger Blume*

Institute of Microelectronic Systems, Leibniz Universität Hannover
Appelstr. 4, 30167, Hannover, Germany,
emails: {guipava,burg,blume}@ims.uni-hannover.de

**ABSTRACT**

Dynamic Partial Reconfiguration (DPR) can be used on Xilinx FPGA devices (by using the internal ICAP unit) to increase the processing performance during run-time and to reduce the hardware cost requirements of a specific system. This paper presents a dynamically self-reconfigurable VLIW-SIMD soft-processor architecture, that takes advantage of the fast reconfiguration of small functional units (FU) instead of hardware accelerators. The proposed soft-processor implements a DMA-ICAP controller to control the dynamic reconfiguration process of its reconfigurable FUs. The implementation of this DMA-ICAP controller together with the software program control are the main topic of this paper. For video-based processing algorithms running under real-time conditions (i.e., 30fps), specialized FUs with equivalent hardware cost of a typical 64-bit SIMD-arithmetic unit can sequentially be reconfigured up to 720 times per frame. Therefore, the use of reconfigurable specialized FUs allows a fine-grained acceleration of intra-frame processing tasks.

## 1. INTRODUCTION

Over the past years, embedded systems have expanded to cover a wide variety of applications, ranging from portable multimedia devices to sensor networks and medical imaging systems. Stringent computing performance and power-consumption requirements in combination with the increasing demand for low cost and time-to-market products make the research field of embedded systems challenging.

In order to meet the mentioned design goals, specialization is required. This can be done by inserting special instructions to a processor or by implementing dedicated hardware macros. One implementation form, which is frequently used, is an application specific integrated circuit (ASIC). Unfortunately, the functionality, flexibility and processing performance of the resulting ASIC remain fixed after the production. This means that in order to adapt these architectures to future changes, the architecture eventually must be physically re-implemented, resulting in long time-to-market and high production costs.

Instead of implementing ”static” ASICs, the TUKUTURI project at the Institute of Microelectronic Systems studies the use of reconfigurable devices to avoid the necessity of completely re-implementing the embedded multimedia processing system physically every time a new enhanced functionality is required. Therefore, dynamic partial reconfiguration (DPR) [1] is used. DPR is provided by commercial Xilinx FPGA devices and can be used to specialize the instruction-set and some parts of the architecture to efficiently execute any kind of multimedia tasks. The time required to reconfigure a specific region of an FPGA directly depends on the size of this region and highly influences the reachable processing performance. For example, in video-based processing algorithms like those used in driver assistance systems [2] (depending on the required reconfiguration time) DPR can be used during inter- or/and intra-frame processing. A usual real-time constraint in this kind of systems is to process at least 25 frames per second (fps), i.e., one frame in less than 40 ms.

Most of the publications, concerning DPR, proposed architectures based on reconfigurable hardware accelerators attached to an OCP bus, which can only be used for inter-frame processing acceleration [1]. Only the hardware demonstrator presented in [2] can perform at least two reconfigurations of sequentially working hardware accelerators within 40 ms, allowing intra-frame processing. In order to efficiently accelerate intra-frame processing, the size of the hardware accelerators should be reduced to enable a speed-up of the reconfiguration process.

This paper presents a reconfigurable VLIW-SIMD soft-processor architecture that can dynamically self-reconfigure the data-path instead of hardware accelerators, performing fine-grain acceleration at the application program code execution level. Section 2 introduces the generic VLIW-SIMD soft-processor architecture, called TUKUTURI. The DMA-ICAP controller used for dynamic partial reconfiguration is described in Section 3. In Section 4, an evaluation of the time required to reconfigure different examples of functional units is presented. Finally, conclusions are presented in Section 5.

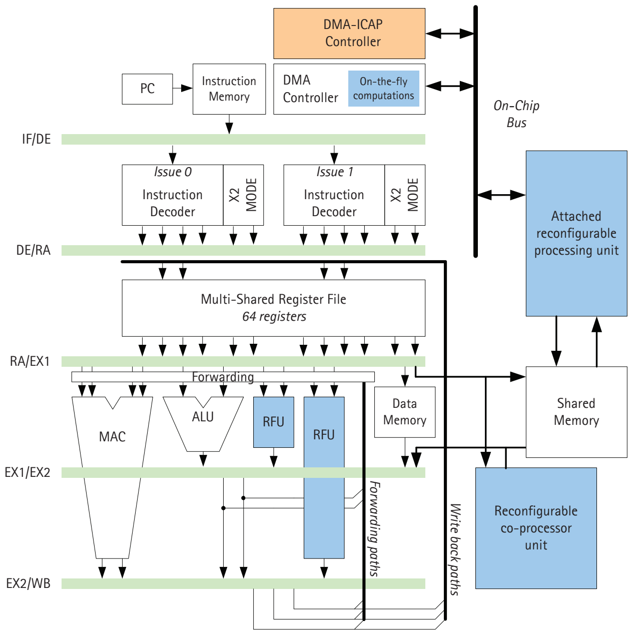

**Fig. 1**. Simplified pipeline scheme of the generic SIMD-VLIW processor architecture (TUKUTURI). The blue shaded areas identify the proposed reconfigurable hardware elements.

## 2. A GENERIC VLIW-SIMD SOFT-PROCESSOR ARCHITECTURE

In previous works carried out at the Institute of Microelectronic Systems in the research project RAPANUI [3], a comprehensive analysis of the architectural design alternatives of application-specific VLIW-SIMD processors for multimedia applications was performed. For that, a comprehensive design space exploration environment based on a generic VLIW-SIMD architecture template was implemented. This environment includes a configurable pipeline architecture simulator, an enhanced assembly code compiler, and a parameterized VHDL implementation of the architecture template. By using this environment, the architecture template can be optimized in terms of performance and hardware cost for a set of multimedia applications. The RAPANUI project has demonstrated that the combination of new enhanced hardware architecture mechanisms and the corresponding assembly code compiler improvements plays an important role to overcome the architectural bottlenecks.

In this work, a new soft-processor architecture, called TUKUTURI (see Figure 1), is proposed based on the mentioned VLIW-SIMD template. The novelty lays in the use of a commercial FPGA device as the destination target platform. The generic VLIW-SIMD processor template was specially designed for efficient processing of macroblocks, commonly used in video coding algorithms, and comprises a flexible data-path controlled by a dual issue-slot VLIW. The control-path is initially divided into 5 basic pipeline stages, but the number of execution stages can be modified to increase the operation clock frequency, since the critical path is located in these stages (when implementing the VLIW-SIMD processor design on a Virtex-5 FPGA). All the SIMD-FUs internally work with 64-bit wide operands, which can be split up to perform the same operation with different data sizes. Finally, it is worth mentioning that the implementation of a basic TUKUTURI configuration based on 5 pipeline stages on a Virtex-5 FPGA can reach more than 100 MHz.

A two-level (re-)configuration strategy is implemented to efficiently use the FPGA resources. First, the TUKUTURI architecture can be optimized to efficiently process a particular application, taking into account the optimal size and number of reconfigurable FUs and reconfigurable co-processor units. Therefore, every time this application is about to be executed, a ***static reconfiguration process*** is performed, programming the whole or a part of the FPGA device before starting the application execution. This level of reconfiguration allows to reuse the system for future applications with totally different processing characteristics, allowing to adapt the complete TUKUTURI architecture to a new application. This mechanism introduces a significant time penalty during run-time, because of the time required to reconfigure those FPGA partitions or the whole FPGA (e.g., a VIRTEX-5 LX330 requires up to 26 ms to be completely reconfigured) and also because of the static reconfiguration (i.e., no computation can be performed on the FPGA during the reconfiguration process). However, the optimization of the TUKUTUTI soft-processor on the architectural level (e.g., number of RFU or static FUs, ...) can significantly increase the processing performance.

Second, ***dynamic partial reconfiguration*** is also supported to insert in run-time new complex instructions or co-processors in the architecture. The proposed RFUs are highly coupled to the pipeline structure of the TUKUTURI architecture, benefiting from the data forwarding-path. The effective use of the forwarding mechanism is crucial for increasing the reachable processing performance. Dynamic reconfiguration will also be supported in other parts of the TUKUTURI architecture, such as the DMA controller and the register file structure. For example, it is planed that the DMA controller supports the use of simple data transformations that will be performed on-the-fly while accessing data from the external memory, e.g. a realignment process proposed in [4].

## 3. THE DMA-ICAP CONTROLLER

DPR is supported by the TUKUTURI processor by means of RFUs and (if required) reconfigurable co-processor units.

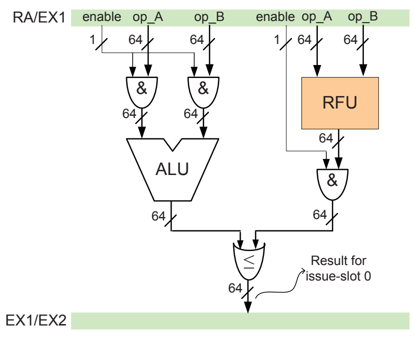

**Fig. 2**. Simplified scheme of a part of the TUKUTURI execution stage.

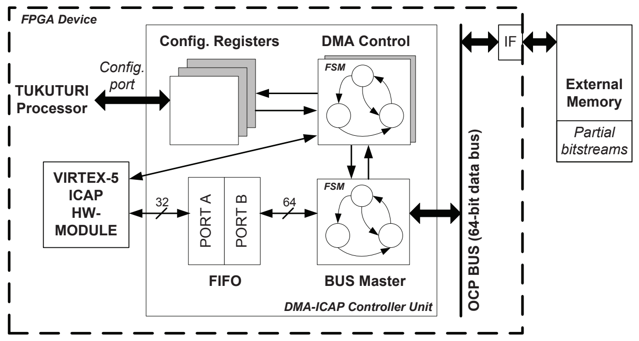

**Fig. 3**. Block diagram of the DMA-ICAP controller interconnected with the TUKUTURI processor and an external memory, where the partial bitstreams are stored.

The RFU coupling mechanism is tighter than the other mechanisms based on hardware accelerators. Moreover, it does not only implement a direct connection between the RFUs and the register file but also supports data forwarding. During the reconfiguration process, the RFU being reconfigured can not be used by the TUKUTURI processor and the outputs of this unit are not propagated, as usual in any pipeline processor. This mechanism is sufficient to isolate the RFU from the rest of the processor architecture.

In Figure 2, the technique used to isolate the RFU is shown. The main concept is that each FU or RFU should generate a zero output if not used. On the one hand, an AND-gate logic is used in non-reconfigurable FUs to produce a zero result in case this FU is not used (i.e., enable signal is zero). This technique was used before in ASIC implementation to reduce the switching activity of the FUs and, therefore, the dynamic power consumption. In FPGA implementation, during the logic synthesis, these AND-gates are distributed inside the FU logic with a marginal increase of the hardware resources. On the other hand, an AND-gate logic is used to control the results of the RFUs. In contrast to non-reconfigurable FUs, this logic will not be distributed inside the RFU logic during the FPGA logic synthesis and it is used to isolate the RFU during its reconfiguration. Finally, an OR-gate logic is used to generate the result operand for each issue-slot.

It is worth mentioning that an efficient FPGA implementation of the SIMD-FUs is crucial for reducing the hardware resources requirements and, therefore, decreases the number of cycles required for reconfiguring the unit [5].

DPR is performed by using the internal ICAP [6] module available in the Xilinx Virtex FPGA family. The TUKUTURI architecture implements a DMA-ICAP concept (see Figure 3) that allows the processor core to read the information required for reconfiguring the FPGA device (i.e., partial bitstreams) from an external memory. The transmission of the partial bitstream and the following reconfiguration process is initiated by the TUKUTURI processor by programming the configuration registers (see Table 1).

Figure 4 shows an example of an assembler code that indicates how the partial reconfiguration works. This assembler code is executed on the TUKUTURI processor to program the DMA-ICAP controller, which performs the reconfiguration of a RFU during the execution of a subroutine that does not use this RFU.

The assembler code can be summarized in the following points: In lines 2 and 3, two *STOREIL* are used to configure the DMA-ICAP. In this example, the partial bitstream is 0x480C bytes large, as written in BLOCK_SIZE. Then, the address position where the partial bitstream is located (in the external memory) is stored in EXT_ADDR. This STOREIL initiates automatically a block-transfer of the indicated partial bitstream to the ICAP unit. In line 6, a subroutine call and return is performed. This subroutine is located in *L_SUBROUTINE_A*. Register *V0R31* is used to return back to this program position after the subroutine execution. Finally, in lines 9 to 14, a WAIT loop is used to check if the DMA-ICAP controller has finished the reconfiguration process (i.e., the partial bitstream transfer). For that, the STATUS_FLAG is read and compared with the immediate value 0x1. These two operations will be executed until the result of the comparison is zero. The STATUS_FLAG register is automatically initialized to 0 after programming the DMA-ICAP controller (i.e., writing in EXT_ADDR), and then it turns to 1 when the partial bitstream has been transfered completely (i.e, BLOCK_SIZE bytes were transfered).

It is desired that the reconfiguration process is performed in background by the DMA-ICAP controller, allowing the processor core to execute other tasks simultaneously. In Figure 4, *L_SUBROUTINE_A* is executed at the same time that the DMA-ICAP controller performs the reconfiguration. Therefore, it is the task of the software engineer to optimally schedule the reconfiguration process of a new RFU together with the execution of a RFU-independent subroutines.

The DMA-ICAP controller also implements internally a dual true port FIFO with different data width ports (a 32-bit data port connected to the ICAP module and a 64-bit data

**Table 1**. DMA-ICAP Configuration Registers.

| Parameter | Description |
|---|---|
| EXT_ADDR | External memory base address where the partial bitstream is located |
| BLOCK_SIZE | Size in bytes of the partial bitstream |
| STATUS_FLAG | Flag that indicates if the DMA-ICAP is still performing the transfer |

```
// Reconfigure RFU1 by programming the DMA-ICAP
STOREIL    0x211, #0x480C   // Write to BLOCK_SIZE
STOREIL    0x210, #0x600000 // Write to EXT_ADDR

// Execute a subroutine that does not use RFU1
JLR      V0R31, L_SUBROUTINE_A

// Check Reconfiguration before using RFU1
:L_WAIT
LOAD       V0R0, 0x212        // Read from STATUS_FLAG
SUBICS_8   V0R1,V0R0,#0x1     // V0R1=V0R0-0x1
                              // store status flags
                              // (e.g. zero)
BSR      L_WAIT, #0b00000001, #COND_ZERO
                              // Jump to L_WAIT
                              // if subword 0 was zero
```

**Fig. 4**. Example of an assembler code that uses the DMA-ICAP to load the RFU1 and checks if the reconfiguration process has finished.

port connected to the OCP bus via a BUS master). This configuration uses the FIFO as a buffer, allowing to reach the maximum available reconfiguration speed even if the external memory is partially busy. It is worth mentioning that the overall system works with more than 100 MHz. Therefore, for this system clock frequency, the DMA-ICAP module, that also works with 100 MHz, only requires one 64-bit data every two system cycles (i.e., one 32-bit data every system cycle) to feed the ICAP module continuously with 32-bit data from the partial bitstream, reaching the maximum bandwidth of 400 MB/s.

## 4. EVALUATION

In case of processing a video sequence (e.g., for video-based driver assistance systems or video coding purposes) under real-time conditions (i.e., 30 fps), the TUKUTURI architecture can sequentially reconfigure a typical 64-bit SIMD arithmetic unit up to 720 times per frame (see Table 2). Moreover, more complex units, such as the disparity map unit presented in [7], can be reconfigured up to 98 times per frame, also allowing the use of DPR in intra-frame video processing. This FU computes the position of the minimum 8-bit value on two vector registers, each one with 32 values. However, there is a technological limitation on Xilinx FPGA devices. For example, on Virtex-5 FPGA devices, only one reconfiguration can be performed simultaneously, although these devices have two ICAP units.

**Table 2**. Time required to reconfigure different representative SIMD-Functional Units on a Xilinx Virtex-5 LX330.

| SIMD Unit | LUTs[1] | RB[2] | Cycles[3] | RPF[4] |
|---|---|---|---|---|
| ClipMaxMin | 170 | 2 | $\sim$ 3080 | $\sim$ 1082 |
| Arithmetic | 293 | 3 | $\sim$ 4620 | $\sim$ 721 |
| Shift-and-Round | 1504 | 12 | $\sim$ 17920 | $\sim$ 186 |
| Disparity-Map[5] | 2835 | 22 | $\sim$ 33870 | $\sim$ 98 |

[1]Look-up-Tables. [2]Virtex-5 Reconfigurable Blocks. [3]Number of cycles required to reconfigure each SIMD-FU. These cycles are measured on a CHIPit emulation system using the DMA-ICAP and the TUKUTURI soft-processor at 100MHz. [4]Number of times that each unit can be reconfigured per frame for a 30 frame per second system constraint. [5]Implementation of a complex SIMD disparity map unit presented in [7] as *DISP_X4*.

## 5. CONCLUSION

In dynamically reconfigurable soft-processors for video signal processing, the size of the reconfigurable modules constraints the maximum number of reconfigurations per frame while processing a frame under real-time conditions (e.g., 30 fps). The evaluation presented in this paper shows that the use of specialized reconfiguration functional units can be used for speeding-up intra-frame processing due to their fast reconfiguration.

## 6. REFERENCES

[1] K. Papadimitriou, *et al.*, "Performance of Partial Reconfiguration in FPGA Systems: A Survey and a Cost Model," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 4, no. 4, pp. 36:1–36:24, Dec. 2011.

[2] C. Claus, *et al.*, "Towards Rapid Dynamic Partial Reconfiguration in Video-Based Driver Assistance Systems," in *Reconfigurable Computing: Architectures, Tools and Applications*, 2010, vol. 5992, pp. 55–67.

[3] G. Payá-Vayá, "Design and Analysis of a Generic VLIW Processor for Multimedia Applications," Ph.D. dissertation, Leibniz Universität Hannover, Feb 2011.

[4] G. Payá-Vayá, *et al.*, "An Enhanced DMA Controller in SIMD Processors for Video Applications," in *Proc. of the 22nd Int. Conf. on Architecture of Computing Systems (ARCS09)*, vol. 5455, 2009, pp. 159–170.

[5] S. Nolting, *et al.*, "Optimizing VLIW-SIMD Processor Architectures for FPGA Implementation," in *ICT.OPEN 2011 Conference*, November 2011.

[6] *Virtex-5 FPGA Configuration User Guide*, Ds202 (v5.3) ed., Xilinix Inc., May 2010.

[7] G. Payá-Vayá, *et al.*, "VLIW Architecture Optimization for an Efficent Computation of Stereoscopic Video Applications," in *Proc. of the 1st Int. Conf. on Green Circuits and Systems (ICGCS2010)*, Jun. 2010, pp. 457–462.