System Level Design for Embedded Reconfigurable Systems using MORPHEUS platform

Paul Brelet, Arnaud Grasset, Philippe Bonnot
Thales Research and Technology
Palaiseau, France
{firstname.surname}@thalesgroup.com

Frank Ieromnimon, Dimitrios Kritharidis
Intracom S.A. Telecom Solutions
Athens, Greece
{fier, dkri}@intracom.gr

Nikolaos S. Voros
Technological Educational Institute of Mesolonghi
Dept. of Telecommunications Systems & Networks
Mesolonghi, Greece
voros@teimes.gr

Abstract— **This paper presents a novel approach for designing embedded reconfigurable systems. It presents the MORPHEUS reconfigurable platform and associated toolset and how they can be used in practice for the development of advanced reconfigurable systems. The paper presents also implementation results from three different application domains and exhibits how MORPHEUS platform can be used for shortening the development time of such systems.**

Keywords – reconfigurable computing; Systems-on-Chip; heterogeneous architectures; dynamic reconfiguration; toolset; embedded systems

I. INTRODUCTION

Taking into account the nature of modern embedded systems and their application domains, we can deduce that two of the most important requirements are cost effectiveness and flexibility. This means that the processing components used in such systems must provide high density performance, enable inexpensive development and be flexible enough so as to change from time to time.

In that context, the approach presented in this paper comes from the need to define a reference platform for dynamic reconfigurable computing to be efficiently used in different application domains. In particular, real-time processing is in the focus of this platform. It is obvious that flexibility, modularity, and scalability of such a platform are key requirements in order to allow an efficient adaptation of the platform architecture to the specific requirements of a certain application.

Another fundamental and new idea is the integration heterogeneous, reconfigurable computation engines (HREs), which support different but complementary styles of reconfigurable computing, in one platform. Three state-of-the-art dynamically reconfigurable computation engines, representing fine-grain, mid-grain, and coarse-grain reconfigurable computation architectures, have been selected and integrated into the MORPHEUS platform.

In summary, the goal of the proposed platform architecture and the associated toolset presented in this paper is to combine the benefits of the different styles of reconfigurable computing in one platform. Since the platform is designed to be highly flexible and scalable, different applications from various application domains can be addressed with it.

The rest of the paper is organized as follows: section II presents an overview of the MORPHEUS platform architecture; section III exhibits the characteristics of the MORPHEUS toolset; section IV presents the experimental results from using MORPHEUS platform and associated toolset in three different case studies; finally, section V presents the paper conclusions.

II. MORPHEUS PLARFORM

The MORPHEUS architecture is based on an ARM9 embedded RISC processor, which is responsible for data, control and configuration transfers between all resources in the system, memory, IO peripherals, and a set of heterogeneous reconfigurable engines (HREs) each residing in its own clock domain with a programmable clock frequency. As dynamic reconfiguration might impose a significant performance demand for the ARM processor, a dedicated reconfiguration control unit is foreseen to serve as a respective offload-engine. Figure 1 depicts the proposed system architecture. All system modules are interconnected via multilayer AMBA busses. Each HRE is composed of a reconfigurable IP seen as a memory-mapped co-processor or peripheral.

Additionally, a NoC infrastructure has been added to extend inter-HRE communication capabilities. It will help to speed-up data delivery among HREs and enable stream data processing, thus removing a bottleneck for parallel data transmission.

MORPHEUS is built around three heterogeneous,

978-0-7695-4076-4/10 $26.00 © 2010 IEEE
DOI 10.1109/ISVLSI.2010.13

IEEE computer society

reconfigurable engines (HREs) which target different flavours of reconfigurable signal processing: XPP-III [7] is a Coarse Grained reconfigurable stream processor featuring an array of 16-bit computational elements communicating through a matrix of configurable data channels. It mainly targets streaming applications with huge computational densities and regular dataflow structures; DREAM [9] is a reconfigurable processor composed by a RISC processor coupled with a mid-grain reconfigurable datapath. DREAM is aimed at exploiting instruction level parallelism for a wide range of applications (e.g. multimedia, telecom, cryptography); FlexEOS [8] is an embedded Field Programmable Gate Array (eFPGA). It is suitable for fine grain algorithm or arbitrary logic implementation. One specific target of the device is the mapping of peripherals due to its 30 GPIO pins exported to the pad frame.

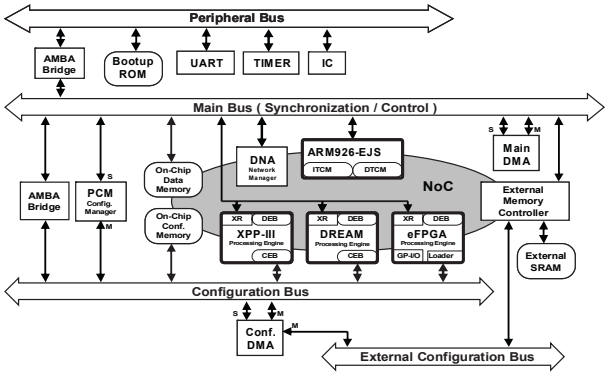

Figure 1. SoC hardware architecture

HREs have been chosen and sized in order to provide a good tradeoff between hardware resources and performance for the selected reference applications, but the basic frame of the design was conceived in order to support additional HREs (e.g ASIC accelerators or DSP cores) or different sizing of the proposed ones. The main design challenge resides in the definition of the top-level architecture, notably the way that data are moved in order to appropriately feed all the computation resources in the system. Differently from ASIC accelerators, that can be switched off when unused, given the area/power cost of HREs such as FPGAs and CGRAs, it is mandatory that all resources are kept active for as long as possible. So the interconnect system must be efficient enough to support the data bandwidth necessary to avoid resource starvation as well as flexible enough to support different interconnect streams depending on each application requirements.

Again differently from hardware acceleration, reconfigurable fabrics need to be programmed, and often feature specific languages and programming styles that require deep knowledge of inner hardware details to ensure good performance. On the other hand, as it is the case with advanced DSP processors or commercial FPGAs, it is often not necessary for the end user to go into the details of the deployment of the single acceleration design that can be written by, or deployed with the close support of the reconfigurable fabric vendor.

III. MORPHEUS TOOLSET

The toolset is able to reduce the application design time and improve the code quality. Shortening the application design time is targeted via the utilisation of high level programming solutions for the platform. Code quality is increased thanks to code generation. Code quality is considered here as code readability, absence of bug with respect to the system level implementations (address mapping, etc), and the possibility to easily implement upgrades and fixes as maintenance.

The MOLEN concept [1] is a way to target this objective since it permits to get abstraction of the tedious management of configurations, execution calls, data communications and process synchronisations. For example, when implementing an application on the architecture, the designer splits its application into control parts (executed on the ARM processor) and computation intensive parts (mapped on the HREs) by following these steps.

Firstly, the application is written in standard C language. During this step, one can validate the application against use cases and test benches.

Secondly, the programmer identifies the functions that must be accelerated by setting a pragma in the application C code on the top of each of them (Figure 2).

Thirdly, these accelerated functions are captured inside a graphical environment, called SPEAR [2], by connecting and assembling some building blocks, called elementary sub-functions and written in C language.

And at the end, the programmer chooses the mapping, it means the programmer selects in which HRE the accelerated function will be launched and after that, the bitstream of the accelerated function for the selected HRE is generated.

```
int pin[10], pout[10];

#pragma MOLEN 1
void func(int*in, int*out){
   *out=*in++;
}

int main(int argc, char*argv[]){
   func(pin,pout);
   return 0;
}
```

Figure 2. Annotation of application code

The high level synthesis of the configurations in the "Spatial Design" part of the toolset whose objective is to provide a common programming interface for the different reconfigurable units also permits to target this objective. These two means notably correspond to masking the heterogeneity of the architecture.

The toolset indeed contributes to get a flexible platform by enabling easy reconfiguration management as stated above and also by enabling run-time dynamic allocation of function to the various reconfigurable units.

Finally the toolset contributes to the performance of the platform by quickly generating optimised executable code. Optimised scheduling shall be performed at compilation. Here, configuration, execution and communication are sources of possible performance optimisation at system level in a complex platform as MORPHEUS is. The high level synthesis of configurations is also a way to get quickly relatively good performances notably by combining the implementation on reconfigurable units and the communication aspects.

A Control Data Flow Graph (CDFG) format is used as an intermediate and technology independent format inside the framework. High-level synthesis techniques are used in a tool, called MADEO [3]. The flow then relies on HRE's proprietary tools to generate the configuration bitstreams. Figure 3 shows a view of the toolset internal tool modules Integration and it demonstrates the central role of the Spacial Design environment which permits to deal with different kinds of capture tools provided they generate a CDFG that is compliant with the defined format. Spacial Design environment also enables the utilisation of several targets: FlexEOS, DREAM and XPP-III.

The user only use the Spatial Design Environment; it is composed by C code and abstracted Models of each functions, and the Makefiles to generate the bitstream for each HREs. In this way, operations can be specified at a high-level of abstraction (improving design time and flexibility), without sacrificing performances. Operations are modeled as a directed acyclic graph, in a formalism called Array-OL which is well suited to represent deterministic, data intensive and data-flow applications such as the kind of operations accelerated on HREs. SPEAR automatically generates a CDFG model of the accelerated function and some communication parameters to feed the HRE. They are forwarded to the SW compilation flow, contributing in this way to a seamless hardware/software integration. The Cascade tool is used to generate the CDFGs for the elementary functions.

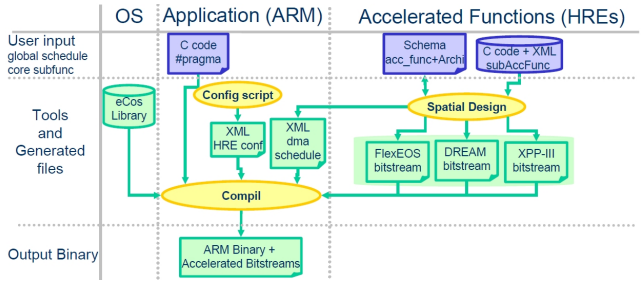

Figure 3. MORPHEUS tool chain

The required effort to implement the application on the MORPHEUS platform is approximately of 3 weeks. In comparison, the measured effort to implement the same application on a SIMD platform is of 6 weeks, and the effort in the case of a classical hardware implementation on FPGA is estimated to 18 PM [5].

IV. Experimental Results

A. Wireless communication case study

The first case study implemented using the proposed approach come from the wireless communications domain. Given the complexity of the specific application, part of it has been implemented using DREAM and FlexEOS) HREs. The rationale for selecting the specific HREs lies a) on the size of the blocks implemented and b) on the processing nature of these blocks.

DREAM was selected to implement a word-level processing block (128-point FFT) of the wireless application, followed by a QAM symbol demapper, capable of supporting modulation schemes ranging from QPSK to QAM64. The exact application scenario is depicted in Figure 4.

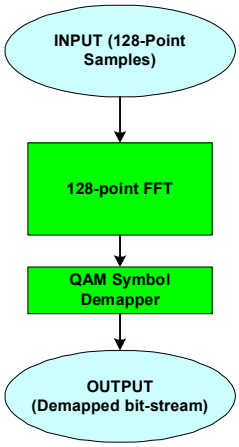

Figure 4. Application scenario for wireless telecommunication case study

As can be seen in Figure 4, the application is a cascade of the above two blocks. Due to the size constraints regarding the FFT block, it was decided to employ the mapping procedure for the creation of the accelerated functions in two discrete steps.

Thus, the SPEAR tool was used for capturing the target application, in two parts: one GDFG was generated for the FFT block and one for the QAM demapper. The resulting CDFGs were used by the MADEO tool downstream for the generation of the configuration streams for the DREAM array, as well as the control code running on the ARM processor, responsible for downloading the configuration streams and for managing the data transfers to and from the DEBs used for communication.

A number of discrete simulation scenarios were run on the PC hosting the tool chain. Simulated execution time can be broken-down to DMA configuration and transfer time, bitstream load/deload time and actual execution time. As can be seen in Table I, the ratio of actual execution time for the ARM+DREAM complex over the corresponding time for the application running on ARM alone is generally favorable to MORPHEUS, indicating speedups ranging from 2.07 to 6.55. The results for the 128-point FFT are particularly favorable,

indicating a speedup of 588. However, if initialization, configuration and data move times are taken into account, the picture is less favorable: overall execution times are degraded by a figure that is ranging from 1.31 to 5.18. Exceptions to this are the figures for the Slicer+QAM64Demapper and of course the 128-point FFT. The latter figure is still a positive speedup of over 45, which is to be expected, since the large volume of computation out-weights the overhead for configuration and data-transfer. For the smaller components evaluated in the simulations though, the results discussed above have certain implications. It seems that the HW-accelerated functions that are taken off the ARM execution environment must correspond to a minimum amount of computation before two conditions can be met: (a) The configuration/DMA/bitstream load/deload time, added to the actual execution time must not exceed execution time on the ARM alone; (b) Reconfiguration intervals must be spaced sufficiently far apart for the configuration/DMA overhead to be absorbed by the gains in execution time.

TABLE I. SPEEDUP RATIOS FOR VARIOUS TEST PROGRAMS VS MORPHEUS CONFIGURATIONS

APPLICATION CASES	ARM+DREAM/ ARM ONLY	ARM+DREAM/ DREAM	ARM ONLY/ DREAM	Total ARM+ DREAM/ Total ARM only
Slicer+De-mapper QPSK	0.2396	1.0887	4.5431	2.8490
Slicer + De-mapper16QAM	0.2163	0.7258	3.3551	1.3173
Slicer + De-mapper64QAM	0.1848	0.8073	4.3890	0.7937
FFT-128	0.0017	1.2704	767.5352	0.0221
Slicer QPSK	0.4098	1.5220	3.7143	4.1756
Slicer QAM16	0.1802	0.4249	2.3575	1.7160
Demapper QAM16	0.1955	0.9151	4.6812	2.5941
Slicer QAM64	0.2060	0.3773	1.8313	1.7192
Demapper QAM64	0.2680	0.5860	2.1866	3.0206

Among the two conditions, the first is almost trivial to meet, provided that the dataset processed by the accelerated functions is large enough. However, an additional condition that must hold is that the HW accelerated functions are static, i.e. there is no reconfiguration at runtime.

If reconfigurability is a desired feature of the application mapped onto MORPHEUS, then the second condition must hold. In the case of wireless case study, the intervals that would make reconfiguration a viable proposition do not seem to exceed a few symbols' worth of data. Thus, it appears that MORPHEUS can indeed support applications that include dynamically configurable components, provided certain timing constraints are met. These constraints are variable, i.e. dependable on each particular application.

B. Smart camera application case study

An intelligent camera can be viewed as a large collection of real time algorithms which are activated (or not) in function of non predictable events such as the content of the image or an external information or a request from the user.

In our case, this image processing application is a motion detection application. At the beginning, this application starts with the absolute pixel-to-pixel difference between the current frame and the background which is periodically refreshed (not considered here). Even if this differentiation allows to isolate the object under analysis (if any), too many details are present as the complete greyscale. For that, a binarization is applied to the frame thanks to a threshold conventionally fixed to 0.3x the maximum value of the pixels.

A "cleaning" task is accomplished by the opening phase, implemented by two operators: the erosion and the dilatation which consist in replacing the central pixel of a 3x3 matrix by the minimum and the maximum values of this matrix.

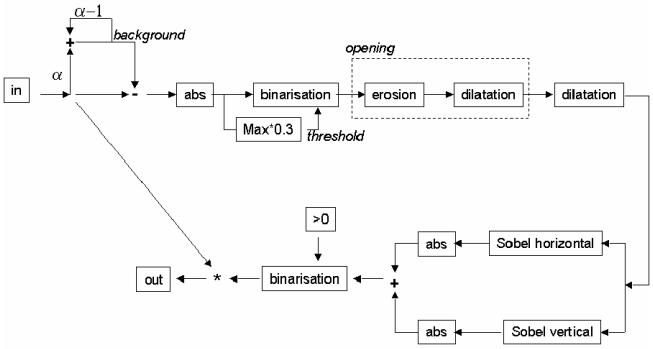

Figure 5. Combination of operators for a simple motion detection algorithm

The next step is the edge detection which allows the identification of the boundaries of the moving shape in the monitored area. This operation is implemented by a simple convolution applied to 3x3 pixel matrices using the Sobel algorithm. Here the Sobel edge detector uses two convolution kernels to detect vertical and horizontal edges.

The resulting image is then binarized, since the aim of the application is not to detect the magnitude of the gradient but the presence of a gradient. Finally the detected edge is merged with the original image. For that goal, inverse binarization is applied: the background is filled by 1s and moving image edges by 0s, thus allowing to implement the merge operation with a multiplication.

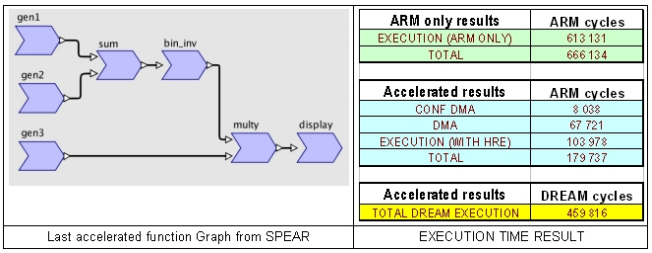

Figure 6. Spear Graph with results

The Figure 5 shows the different steps of the image processing application for an intelligent camera.

All the application is generated by the toolchain in 4 accelerated functions: the first accelerated function corresponds to the subtraction, the absolute value and the binarisation between two images (the background and the current image); the second accelerated function is the opening (erode+dilate) and a second dilate; the third accelerated function corresponds to the convolution H/V from a SOBEL matrix; the last accelerated function is the sum, the absolute value and the multiplication between the current image and the result of the third accelerated function.

Figure 6 was obtained by manual implementation of the application on the DREAM unit only (the DREAM target frequency within MORPHEUS is 250 MHz). The measured performance shows a speedup of x3.7 with the utilization of the DREAM.

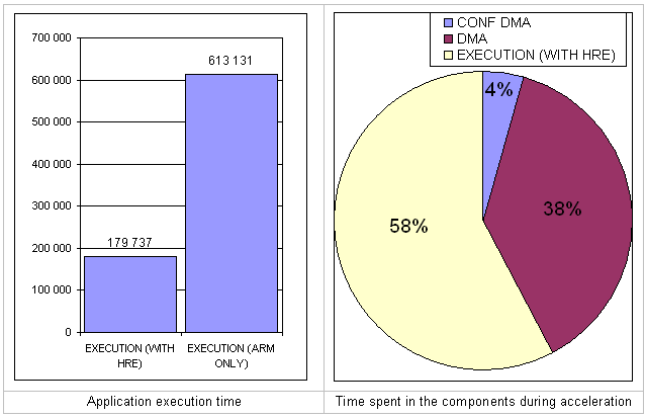

Figure 7. Execution time

By using the different HRE available on the MORPHEUS platform, the performance could even reach 1.27 cycles per pixel (20% performance increase). For this implementation, the critical kernel is the Erosion/Dilatation/Edge Detection. It has been implemented on the DREAM engine whose frequency is mentioned above.

Figure 7 shows the repartition about the time spent by the component during the execution. The DMA communications take about the same time as the HRE execution. The DMA transfers can be improve in using the pipeline stage to decrease the time of computation by the HRE. In a case where the DMA and the HRE work would take place in parallel (e.g. when implementing the pipeline) we would gain about half the total time (i.e. the DMA time would then no linger be added to the computation time at least in the pipeline).

Moreover, the higher level programming that is allowed by the toolset gives a programming accessibility to a larger range of programmers. Since it is not needed to get specific skill corresponding to a knowledge about low level architectural considerations, even people not really aware about the platform characteristics can program the chip.

C. Discrete wavelet transformation application

The wavelet transform was borne out of a need for further developments from Fourier transforms.

Discrete wavelet transforms (DWT) are applied to discrete data sets and produce discrete outputs. Transforming signals and data vectors by DWT is a process that resembles the fast Fourier transform (FFT), the Fourier method applied to a set of discrete measurements.

The DWT is being increasingly used for image compression today since it supports features like progressive image transmission (by quality, by resolution), ease of compressed image manipulation, region of interest coding, etc.

In our case, the DWT is applied for an image processing application composed by FIR filter bank structures.

The architecture of the MORPHEUS demo is presented in full detail in [6].

The 2D-DWT transform is repeated 3 times (for the 3 levels of compression). Each level is made of vertical and horizontal filtering. In each direction a low pass filter and a high pass filter are computed.

The C code for the high pass filter and the low pass filter has been written. These two sub-functions are sufficient for the capture of the whole 2D-DWT transform as shown in the SPEAR capture figure. This application is composed by four main accelerated functions (Figure 8): The first function is a Filter combination (High-pass Filter + High-pass Filter); the second accelerated function is also a combination of Filter (High-pass Filter + Low-pass Filter); the third accelerated function is a Low-pass Filter and a High-pass Filter; the last accelerated function correspond to Two Low-pass Filters.

To simulate the DWT application, we have resized the initial image 5200*5200 (High Definition Image) to a image 80*80 (Low Resolution).

The code for each Filter functions is very simple, and each filters has been separated during the test. Their implementations on the HREs take less than a day effort for each of them and do not require any HREs knowledge. Only the C code is required. The design on the SPEAR graphical interface can easily be built and requires a few minutes. Globally, the implementation of the application including C code for ARM and SPEAR capture take a few days. The application can be implemented within approximately 4 days on the MORPHEUS platform where the non-integrated approach requires several weeks.

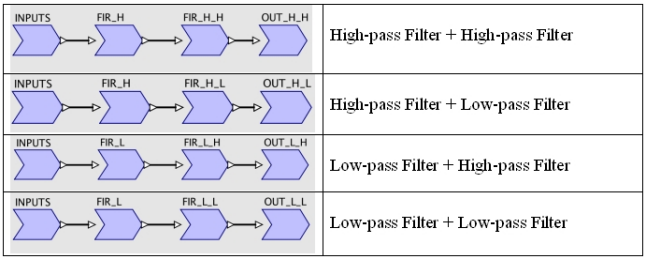

Figure 8. Discrete wavelet transformation decomposition

The high pass and low pass filters composing the 2D Discrete Wavelet Transformations have been implemented on the DREAM and the eFPGA. Figure 9 shows the implementation results. In our case studies, these filters are

applied on an array of 80 x 80 integers.

	High-pass Filter + High-pass Filter	High-pass Filter + Low-pass Filter	Low-pass Filter + High-pass Filter	Low-pass Filter + Low-pass Filter
ARM only results	**ARM cycles**	**ARM cycles**	**ARM cycles**	**ARM cycles**
EXECUTION (ARM ONLY)	57 025	50 725	49 805	49 503
TOTAL	66 687	60 387	59 467	50 304
Accelerated results	**ARM cycles**	**ARM cycles**	**ARM cycles**	**ARM cycles**
TOTAL CONF_DMA	6 025	5 948	5 947	5 947
TOTAL LOAD/DELOAD	5 689	5 699	5 884	5 880
EXECUTION (WITH HRE)	6 404	8 749	11 713	13 051
TOTAL	27 738	30 016	33 164	35 325

EXECUTION TIME RESULT

Figure 9. Discrete wavelet transformation results

For this application, the complete tool flow has been involved (compilation, RTOS, Spatial Design). Simulations on the MORPHEUS SystemC platform have been run and provided correct results. Simulations have been performed on the complete chain except the prefix removal for the moment. Figure 10 shows the performance given by the MORPHEUS toolset for the DWT image processing application, there are in the same order than in the previous case (4x gain).

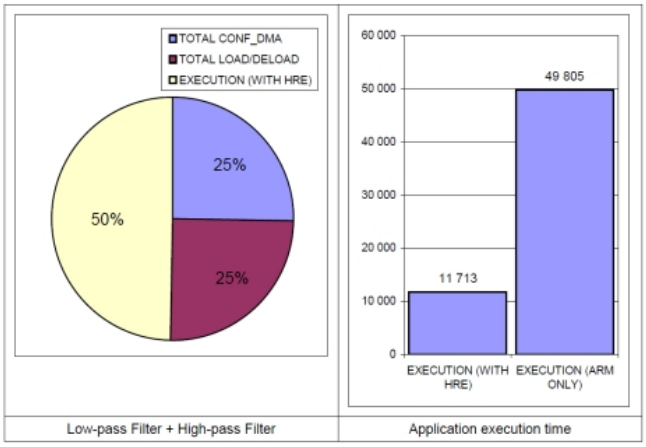

Figure 10. Execution time

Regarding the development effort, the implementation time of the film grain removal algorithm on the MORPHEUS platform was approximately the same as the implementation time of the algorithm on a state of the art FPGA-based platform by opposition with the utilization of the MORPHEUS toolset, moreover, productivity gains have been observed (x4) as well as programming easiness thanks to higher level programming. However, in this case, the DMA configuration and the LOAD/DELOAD times are equals to the execution times on a HRE (e.g. the pipeline stage is not implemented), nevertheless, the MORPHEUS toolset obtains good results in comparison with a manual SIMD program.

V. CONCLUSION

In the previous sections we presented the concept of a novel heterogeneous platform, called MORPHEUS, and the associated toolset for the design of reconfigurable embedded systems.

The design of embedded reconfigurable systems using the MORPHEUS platform starts using the toolset presented in the previous section. The initial system description is usually in C code and the toolset generates accelerator target specific codes and optimises the schedule of the application implemented.

The feasibility of the proposed design approach has been justified through three different case studies coming from complementary domains. In all three cases it has been proven by implementation results that the MORPHEUS approach enables the design of embedded reconfigurable systems in shorter development times compared to traditional design flows of embedded reconfigurable systems.

REFERENCES

[1] VASSILIADIS, S., WONG, S., GAYDADJIEV, G., BERTELS, K., KUZMANOV, G., and PANAINTE, E., 2004 The MOLEN polymorphic processor. *IEEE Transactions on Computers, vol. 53, no. 11*, 1363–1375.

[2] LENORMAND, E., AND EDELIN, G. 2003. An Industrial Perspective: Pragmatic high-end signal processing environment at Thales. In *Proceedings of the 3rd International Workshop on Synthesis, Architectures, Modeling and Simulation* (SAMOS).

[3] CAMBONIE, J., GUÉRIN, S., KERYELL, R., LAGADEC, L., POTTIER, B., SENTIEYS, O., WEBER, B., YAZDANI, S. 2004. Compiler and system techniques for soc distributed reconfigurable accelerators. In *Proceedings of the 4th International Workshop on Synthesis, Architectures, Modeling and Simulation* (SAMOS), 293-302.

[4] FABIO CAMPI, ANTONIO DELEDDA, MATTEO PIZZOTTI, LUCA CICCARELLI, CLAUDIO MUCCI, ANDREA LODI, LUCA VANZOLINI and ARSENI VITKOVSKI, "A dynamically adaptive DSP for heterogeneous reconfigurable platforms" Proceedings of the Design Automation and Test in Europe (DATE), Apr. 2007.

[5] BONNOT, P., LEMONNIER, F., EDELIN, G., GAILLAT, G., RUCH, O., and GAUGET, P. 2008. Definition and SIMD implementation of a multi-processing architecture approach on FPGA. In *Proceedings of the Conference on Design, Automation and Test in Europe* (March 10 - 14, 2008). DATE '08, pp. 610-615.

[6] PUTZKE-RÖMING, W., 2009. MORPHEUS Architecture Overview. In *Dynamic System Reconfiguration in Heterogeneous Platforms - The MORPHEUS Approach*, vol. 40 of series Lecture Notes in Electrical Engineering, chapter 3, pp. 31-37, Springer Netherlands.

[7] XPP-III Processor Overview, white paper, PACT XPP Technologies, 2006. Available: http://www.pactxpp.com

[8] M. BARON, "M2000's Spherical FPGA Cores", in MicroProcessor Report, December 2004.

[9] F. CAMPI et al., "A dynamically adaptive DSP for heterogeneous reconfigurable platforms", ", in Proc. of the Conf. on Design, Automation and Test in Europe (DATE'07), 2007.