2009 12th Euromicro Conference on Digital System Design / Architectures, Methods and Tools

xMAML: a Modeling Language for Dynamically Reconfigurable Architectures

Julien Lallet, Sébastien Pillement, Olivier Sentieys
IRISA-INRIA, University of Rennes
Lannion, FRANCE
Email: {lallet, pillemen, sentieys}@irisa.fr

***Abstract*—Constant evolution of norms and applications, usually implemented on system-on-chip (SOC), increases architecture performance and flexibility requirements. Current architectures are consequently becoming more complex and difficult to develop. One of the solutions is to develop design frameworks based on high-level architecture description languages (ADL). These ADLs are useful for a rapid description of the hardware that should be implemented on an architecture. Designers can use ADL for the development of generic front-end tools. Our framework aims at designing dynamically reconfigurable architecture with the help of an ADL. This paper presents xMAML, an architecture description language dedicated to the instantiation of dynamically reconfigurable heterogeneous computing units. From this ADL, a synthesizable model is produced after exploration, simulation and validation phases. As proof of concept, exploration for a WCDMA receiver on two dynamically reconfigurable architectures is presented.**

***Keywords*-ADL; dynamic reconfiguration;**

I. Introduction

New kind of system-on-chip (SoC) architectures are required to face the constant increase in application performance requirements. Moreover, the evolution of norms and applications makes necessary the use of new scalable architectures. In the last few years, the most widespread issue used by researchers was to design new architectures with the help of development frameworks based on high-level architecture description languages (ADL). Currently, ADLs are widely used in the design of processor architectures, and some of them permit to develop homogeneous multiprocessor architectures. These methodologies permit fast design of both applications and architectures. Firstly, ADLs allow fast description of the resources which compose the architecture, and provide information which can be used by front-end tools. These front-ends are generic and support ideally all architectures described by the ADL. Secondly, due to the fast design cycle, an ADL model allows to quickly explore the characteristics of the architectures under design, and to verify the application constraints.

In order to cope with flexibility, SoCs include more and more reconfigurable blocks, allowing longer lifetime of the architectures (by permitting update) and adaptability to new applications (by the way of reconfiguration). One of the main problem comes with the costs overhead due to the configuration process. One solution consists in the use of dynamically reconfigurable architectures in order to reduce the required die size and to improve the architecture usage. The design space of such architectures [2] is very large, and designing a Dynamically Reconfigurable Architecture (DRA) is long and error prone. Applying the ADL based methodologies is then a promising solution. DRAs are parallel architectures composed of arrays of heterogeneous resources (such as logic blocks, memories, processor cores, ...), and can be expressed by some ADLs. Currently, ADLs cannot handle the specific part of DRA, i.e. the reconfiguration process. xMAML (extended MAML) is based on MAML language [1] *(MAchine Markup Language)* which is particularly powerful for the description of massively parallel processor architectures. As an extension of this ADL, we add new concepts in order to support heterogeneous and dynamically reconfigurable hardware resource description. In this paper, we present the ADL xMAML which aims at describing dynamically reconfigurable architecture, hence giving the possibility to explore several architecture models designed with dynamic reconfiguration capability.

The remainder of the paper is organized as follows. Section II describes related works on architecture description languages. Since these ADLs were developed for processor architectures, we present advantages and drawbacks of several approaches. We motivate in this section the choice of our ADL. Section III presents our contribution on the MAML language. More precisely, we present the modifications and extensions developed to support dynamic reconfiguration. Results of the implementation of a WCDMA receiver targeting a FPGA modeled with xMAML and a coarse-grain dynamically reconfigurable processor model are discussed Section IV. Finally, Section V sums up this paper and introduces future works.

II. Related Works

Architecture description languages (ADL) are used for fast modeling of hardware architecture resources such as general-purpose processors (GPP) or specialized processor. They are also used as an input for design and compilation tools, and hence permitting the generation of customized compilers or the generation of simulation and synthesis specifications [9]. Furthermore, depending on the level of abstraction, it is possible to quickly explore power consumption, performances or to simulate the described architecture. Three categories depending on the level of abstraction of the architecture specification can be identified.

Structural ADLs (e.g. MIMOLA [3], UDL/I [10]) consist in the description of architectures mainly at the Register Transfer Level (RTL). This kind of hardware description

978-0-7695-3782-5/09 $25.00 © 2009 IEEE
DOI 10.1109/DSD.2009.151

is comparable to VHDL or Verilog languages. Structural ADLs are useful for concrete structural description of the architecture, which allows to control the quantity of used resources with a good precision. Nevertheless, when the architecture is very complex, the architecture specification becomes difficult, because of the amount of resources to describe. This is also a problem when the architecture has to be explored or simulated. Since a lot of details are represented at this level, simulation time can become very high.

At the opposite, the second category corresponds to behavioral ADLs (e.g. nML [6], ISDL [7]) which enable the specification of an architecture using a set of instruction semantics. The fast description of an architecture is allowed by ignoring the structural hardware details. Designers can then focus on the functionality of the architecture, and the exploration process and simulation performances are hence faster. Nevertheless, when generating the architecture, the amount of hardware resources produced is hard to control. The evaluation and simulation are also less accurate than in structural ADLs since no information on hardware structure are provided.

Finally, mixed ADLs (MAML [1], ARMOR [5]) allow model specifications between behavioral and structural representation. These languages consist in extending the behavioral ADLs with low-level hardware resources specification on some essential parts. Mixed ADLs constitute a trade-off between fast and simple description from behavioral ADLs and precision of the description of structural ADLs. This approach is well suited for the definition of specialized processors that require some specific parts to be optimized, while keeping the evaluation of the whole architecture simple and fast.

To the best of our knowledge, no ADLs have been developed for the specification of heterogeneous dynamically reconfigurable architecture. Some research was previously made on reconfigurable architecture development frameworks which include ADLs specification languages. However, they are either dedicated to FPGAs and do not manage dynamic reconfiguration, such as the MADEO [13] framework or VPR tool [4], or to massively parallel processors, such as the MAML [1] framework. We propose an ADL enabling the description of dynamically reconfigurable architecture models from FPGAs to reconfigurable processor. We believe that a mixed ADL is perfectly suitable for this reconfigurable architecture description.

III. xMAML: ADL EXTENSION FOR DYNAMIC RECONFIGURATION

In order to provide dynamic reconfiguration specification on an ADL, some new concepts have to be introduced. These concepts should enable the specification of

- Partial reconfiguration: as we intend to model dynamically reconfigurable architecture, we should support the partial configuration process;
- Management of heterogeneity: current dynamically reconfigurable architectures are heterogeneous and embed a lot of resources. They are composed of several kinds of resources such as logic cells for hardware acceleration, embedded processor, memory, etc. At this level, we should support the definition of the granularity of the architecture (i.e. fine-grain or coarse-grain architectures);
- Preemption mechanism: new approach in reconfigurable computing intends to manage the dynamic reconfiguration as a preemption of tasks. This corresponds to the preemption of a task described in hardware and will require ad-hoc mechanisms to be usable and realistic [12];
- Management of the number of possible configurations: the dynamic reconfiguration requires to manage several bitstreams.

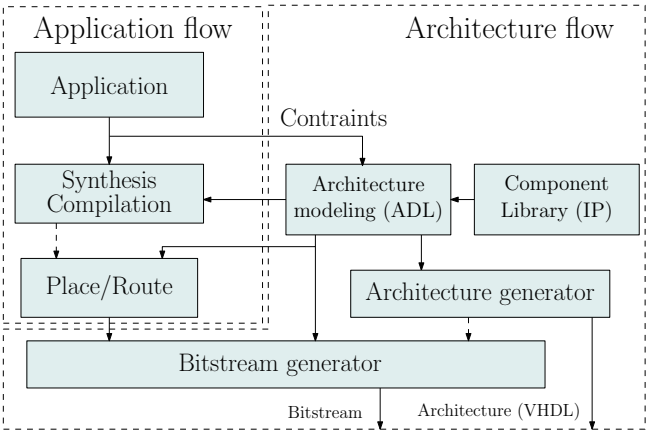

Figure 1. Development process of a dynamically reconfigurable architecture dedicated to an application

Fom our point of view, the design flow of a new dynamically reconfigurable architecture is divided into two flows. First, the architecture design flow on the right (Fig. 1) starts with the high-level description of the architecture from a specification on parameters and constraints of the reconfiguration processes. When the flexibility, dynamicity and performance parameters have been explored, tested and approved, an RTL model can be automatically generated from the ADL. In parallel, the application flow (Fig. 1 left) follows the traditional design flow and generates a configuration bitstream depending on the application to implement and on the target architecture.

A. MAML: basis ADL of xMAML

The MAML language is developed for the description of massively parallel architecture. Therefore, the language includes mechanisms for the description of homogeneous arrays. MAML is used for the description of architectural parameters useful to front-end tools such as synthesizer, partitioning, ordering and task allocation methodology. Furthermore, extracted information from the MAML architecture description can be used for simulation purpose.

The MAML syntax is based on the XML language. A MAML description consists in two levels of abstraction: a

high description level and a register transfer level (RTL).

1) At the RT level, it is possible to specify the complete structure of a processing element (PE). An architecture is then a precise description of the internal resources including functional units, storage elements (control/-data registers, local memories, instruction memories, FIFO, etc.), internal connections, instructions (instruction codes, functionality, SIMD capabilities).
2) The MAML specificity comes from the concepts which have been developed to specify massively parallel processors. Few parameters are needed to automatically produce the whole architecture. For example, it is necessary to specify the size of the processing matrix and the interconnection scheme for all PEs which belong to the same matrix.

All these features allow to rank MAML as a powerful tool for massively parallel processor architecture description. However, this ADL is not adapted to the specification of any other kind of dynamically reconfigurable architectures.

B. xMAML: Extended MAML for Dynamic Reconfiguration

Our contribution to the extension of the MAML languages deals with the specification of dynamically reconfigurable architectures as defined at the beginning of Section III. This is achieved by the introduction of three main concepts.

1) First, the interconnection network has to be flexible enough to allow communications between any kind of computing resources.
2) Secondly, it is necessary to split the architecture into several configuration areas. This allows to partially and quickly reconfigure the architecture.
3) Finally, reconfiguration time and reconfiguration control has to be improved in order to support today's application performance requirement.

In order to optimize the reconfiguration time and to support preemption, a multiple context [8], [11] approach is used as reconfiguration model. The interested reader can refer to [14] for more information on the reconfiguration mechanisms implemented.

A xMAML description of a DRA is possible through the specification of a few basic elements, which allows the specification of a huge variety of architectures. Fig. 2 shows the architecture model composed of I/O units (IO_u), interconnection units (I_u), computing units (C_u) and multi-context units (R_u). Each unit can be specified through the xMAML language and thus be generated as a HDL description. The computing units, also called Processing Elements, can be implemented from a library and are either fine-grained or coarse-grained units.

The flexibility of the interconnection network is achieved thanks to the definition of a new flexible connection resource. A connection resource is the hardware implemented to enable communications between PEs. Indeed, the connection scheme used in MAML is very powerful, but can only be applied for homogeneous parallel processors. This is due to the fact that each PE which composes a matrix has the same connection scheme as the others. This is not the fact

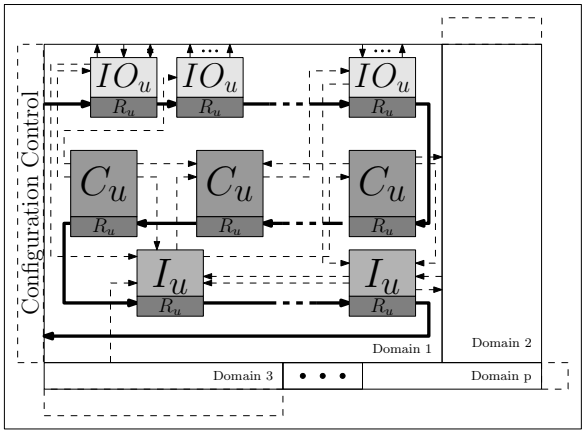

Figure 2. Example of the generic architecture model composed of I/O units (IO_u), interconnection units (I_u), computing units (C_u) and multi-context units (R_u)

for heterogeneous architectures. The new interconnection resource allows the specification of the reconfiguration time, whose value changes the number of necessary resources for the reconfiguration (*ReconfigurationTime*). Then, the port numbers and types of connection enable to extract the necessary features for the routing tools (*DBPorts* and *PElementsPorts*). Finally, the parameter *AdjacencyMatrix* allows restriction on the connection possibility of one communication port. Fig. 3 gives an example of an interconnection resource modeled in xMAML.

```
<PEInterconnectDyRIBox name="DB">
 <ReconfigurationTime cycle="1"/>
 <DBPorts>
  <Inputs number="4" bitwidth="8" />
  <Outputs number="4" bitwidth="8" />
 </DBPorts>
 <PElementsPorts>
  <Inputs number="1" bitwidth="8" />
  <Outputs number="1" bitwidth="8" />
 </PElementsPorts>
 <AdjacencyMatrix>
  <DOutput idx="0" row="01111" />
  ...
  <POutput idx="0" row="11110" />
 </AdjacencyMatrix>
</PEInterconnectDyRIBox>
```

Figure 3. Example of the xMAML description of an interconnection resource

As soon as a computing area resumes the execution of its task, it is important to immediately reconfigure the corresponding resources without disturbing the other computing areas of the architecture. This corresponds to the partial reconfiguration process. xMAML allows the specification of reconfiguration area by the definition of the *domain* concept (Fig. 4). In such a domain, all the processing elements belong together to one reconfiguration path. However, it is possible to merge several domains together in order to compose one reconfiguration area thus allowing the implementation of large tasks. It is important to notice that computing areas and reconfiguration areas are completely independent. A

computing path can cover several reconfiguration domains. Each configuration domain is independent and can be configured alone, while the other domains remain unchanged. The designer has the possibility to specify some parameters in order to prevent the generation of some specific capabilities of the architecture as preemption, partial reconfiguration or interruption. Fig. 4 shows a description of a domain where the concept dedicated to dynamic reconfiguration are specified (*preemption*, *partialReconfiguration*, *confBusWidth*) for the specification of the configuration bus size and *IRPriorityLevel* for the specification of the number of interruption and priority which can be managed by the reconfiguration manager). Furthermore, static communications are specified in the *InternalConnections* between resources instantiated by the *Instantiation* area or *ElementPolytopRange* area.

```
1  <DBDomain name="DPR_1" >
2   <ReconfigurationParameters preemption="disable"
        domainCtrl="shared" partialReconfiguration="
        enable" IRPriorityLevel="3" taskNumber="10"
        confBusWidth="8"/>
3   <Interconnect type="manual" >
4   <Instantiation name="MultBus" instanceOf="DBox"/>
5   <InternalConnections>
6    <MultBus:in(1) = DataMem1:output_0(0:15)/>
7    <MultBus:in(2) = DataMem2:output_0(0:15)/>
8    ...
9    <AG4:output_0(0:15) = DataMem4:input_0(0:15)/>
10  </InternalConnections>
11  </Interconnect>
12  <ElementsPolytopeRange>
13   <MatrixA row = " 1 0"/>
14   ...
15   <VectorB value = " 4"/>
16  </ElementsPolytopeRange>
17 </DBDomain>
```

Figure 4. Example of the xMAML description of a reconfiguration domain

Finally, the specification of the PE interfaces is also needed in order to generate the specific hardware dedicated to the implementation of the dynamic reconfiguration. The specification of the ports dedicated to the reconfiguration are particularly necessary since the ports *ConfigIn*, *ConfigAddr* and *RW* from the description of a logic block of the example of the Fig. 5.

```
1  <PEInterface name="clb">
2   <Reconfiguration cycle="16" bits="16" preemption="no
        "/>
3   <IOPorts>
4    <Port name="luti0" bitwidth="1" direction="in" type
        ="data" />
5    ...
6    <Port name="ConfigIn" bitwidth="1" direction="in"
        type="RAMConfIn"/>
7    <Port name="RW" bitwidth="1" direction="in" type="
        RAMConfEn"/>
8    <Port name="ConfigAdre" bitwidth="4" direction="in"
        type="RAMConfAddr"/>
9   </IOPorts>
10 </PEInterface>
```

Figure 5. Example of the xMAML description of a PE interface

By adding the above properties to the MAML language it is possible to explore the main characteristics of an architecture from the point of view of reconfiguration. We focus only on this point since lot of works intends to explore more classical parameters (such as execution time or area).

C. Exploration on the parameters

From the ADL description, exploration is made easier and faster than using a low-level description. In the context of DRA, we have explored the influence of the dynamic reconfiguration on area, power and timing constraints. The parameters of the exploration handled are the estimation of reconfiguration resources, the reconfiguration area overheads, the reconfiguration time, the estimation of the preemption cost, the management of the reconfiguration. The reconfiguration resource is the hardware needed to store the future context of the computing resource. The underlying architecture model offers a great flexibility on the interconnect structure of the platform. In order to make faster evaluation, we have pre-characterized the interconnect and extracted related costs and performances.

1) Exploration on Flexible Interconnect: Interconnection resources have been synthesized with different parameters such as the data size and the number of possible connection on one output. The influence of these parameters on silicon area (Fig. 6) and power (Fig. 7) are presented. First, for both area and power, the normalized value for data size of 16, 32 and 64 bit-width gives the same results. For large data bit-width, the impact of the implementation of reconfiguration processes is directly proportional to the data size. The normalized value are given in terms of area or power for one bit.

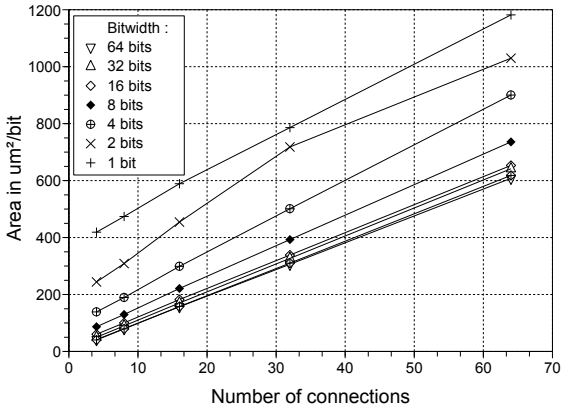

Figure 6. Influence of IO bit-width on normalized area

For smaller data bit-width, the area normalized to one bit is even more significant when the data size is smaller. In conclusion, it is more efficient, from an area point-of-view, to route large data, than to route signals bit by bit. The influence of the number of possible connections on one output increases the silicon area proportionally to the data bit-width. This parameter, which represents the flexibility of the interconnection blocks, has an impact depending on data

size. A trade-off between hardware costs and flexibility has then to be taken into account.

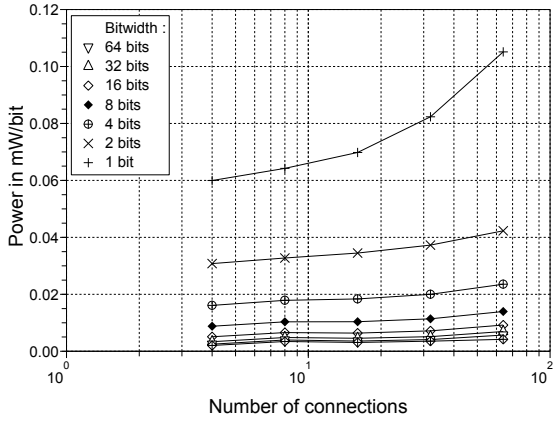

Figure 7. Influence of IO bit-width on power consumption

Power analysis in Fig. 7 shows that, for data bit-width less than 16 bits, the normalized power is even more significant when the the data bit-width is smaller. As for the area, driving large data is more energy efficient. The difference lies here in the power consumption of wide possible connections on one output. Unlike silicon area, exploration shows that, in this case, power is more efficient when less possible connections are computed for larger bit-width.

These results show that a good trade-off between power consumption and silicon area is achieved when the interconnection resources interconnect as few as possible connection and on the largest data bit-widths.

In order to estimate the reconfiguration time, it is necessary to compute the bitstream size required to reconfigure the interconnection resources. Each output of an interconnection resource is controlled by one scanpath register [14] . Its value selects the right multiplexer input to connect to the output. The bitstream size for an interconnection block composed of Nb_{output} outputs is equal to

$$N_{bconfInterco} = \sum_{n=0}^{Nb_{output}-1} \lceil log_2(I_n) \rceil \quad (1)$$

where I_n is the number of inputs that can be connected to output n.

2) Exploration of global parameters: At this level, the main objectives are to determine the size of the configuration data and the number of configuration domains. Choosing the architecture of a processor element (PE) determines the required number of configuration bits $N_{bconfPE}$. Then, the exploration on the interconnect estimates the number of bits required by the interconnect $N_{bconfInterco}$ (eq. 1). The designer, according to his application requirements, specifies the size of the architecture (i.e. the number of PE required). For an architecture composed of N_{PE} PEs and N_{DB} interconnection blocks, the total number of configuration bits is

$$B_s = N_{PE} \times N_{bconfPE} + N_{DB} \times N_{bconfInterco} \quad (2)$$

B_s is the bitstream size in bits of the reconfigurable architecture. This parameter determines the size of the configuration memory required.

Therefore, the time to reconfigure the architecture is equal to

$$t_{reconf} = \frac{B_s}{CBW} \times t_{memory} \quad (3)$$

where CBW is the configuration bus width and t_{memory} is the access time of the configuration memory delivering data of width CBW. CBW is estimated at the same time that the number of domains nD required according to timing constraints. nD is estimated from the time required to reconfigure the whole architecture and the available time between each configuration requirement (t_{aRt}) which is application dependent.

$$nD = \frac{t_{reconf}}{t_{aRt}} = \frac{\frac{B_s}{CBW} \times t_{memory}}{t_{aRt}} \quad (4)$$

For example, the exploration could make possible the automatic area splitting, or the automatic generation of the domains.

By giving a timing constraint, it is easy to determine the reconfiguration bitstream size of the described architecture and to determine the bitstream propagation time. Thus, in case of too restrictive timing constraint, it is possible to split the architecture in as many reconfiguration domains as necessary.

3) Influence of preemption on resource costs: The reconfiguration resources can be used for preemption purpose. That means that it is possible to stop and to extract the current context for a future reconfiguration. This extraction uses the same resources as the ones for the configuration at the cost of a more complex interconnect. Preemption implementation has an impact on silicon costs. The preemption mechanism can be automatically generated from the xMAML description of the architecture. This mechanism includes the use of scanpath registers as in circuits with *Design For Testability*. This hardware solution costs around 10% more resources than a classical register, but allows to extract the contexts at any time.

Preemption requires the same amount of time than the reconfiguration and uses the same resources. Implementing preemption divides the available time between each configuration t_{aRt} by a factor 2.

4) Automatic generation of the architecture: Once each exploration parameter has been validated, the VHDL RTL architecture model is automatically generated which includes both computing resources and reconfiguration control resources.

IV. Case-study

In this section, we present the implementation of a Wideband Code Division Multiple Access (WCDMA) receiver on an embedded FPGA based on the xMAML model of a

Xilinx XC4000 CLB [17] and on the xMAML model of the reconfigurable processor DART [16]. Both implementations will be explored in order to determine the best implementation solution.

A. WCDMA Presentation

WCDMA is a high-speed transmission protocol used in third generation mobile communication systems, and is considered as a complex application. It is based on the CDMA access technique where all data sent within a channel and for a user have to be coded with a specific code to be distinguished from the data transmitted in other channels [15]. The number of codes is limited and depends on the total capacity of the cell, which is the area covered by a single base station. To be compliant with the radio interface specification, each channel must achieve a data rate of at least 128kbps. The theoretical total number of concurrent channels is 128. As in practice only about 60% of the channels are used for user data, the WCDMA base-station can support 76 users per channel. The WCDMA application executed on the reconfigurable architectures consists in the execution of three main tasks: *FIR (Finite Impulse Response) filter*, *Searcher*, *Rake Receiver*. Within a WCDMA receiver, real and imaginary parts of the signal received on the antenna after demodulation and analog-to-digital conversion, $S_r(n)$, are filtered by an FIR *shaping filter*. Since the transmitted signal reflects in obstacles like buildings or trees, the receiver gets several replica of the same signal with different delays and phases. By combining the different paths, the decision quality is drastically improved. Consequently, the *Rake Receiver* combines the different paths extracted by the *Searcher* block in order to improve the quality of the symbol decision. Each path is computed inside the *Rake Receiver* by a *finger* which correlates the received signal by a spreading code aligned with the delay of the multi-path signal. In our case, a maximum number of 6 fingers is considered. The decision is finally achieved on the combination of all these widespread paths. The bandwidth of the transmitted signal is equal to 5 MHz. The frequency of the code corresponding to the chip rate (F_{chip}) is fixed to 3.84 MHz. One slot is composed of 256 chip data. Registers are used to pipeline data while *FIR*, *Searcher* or *Rake Receiver* compute the result for one slot. The computing time available for the three functions is therefore $t_{slot} = 66.6\ \mu s$ between the computation of two consecutive slots. The *FIR* and the *Searcher* requires 1024 samples, while one *Finger* of the *Rake Receiver* computes 256 samples.

In a first section, we will explore the hardware resources needed to implement a dynamically reconfigurable WCDMA decoder on an embedded FPGA, and in a second section, the hardware resources needed for the same application on the dynamically reconfigurable processor DART.

B. Exploration of the implementation on an embedded FPGA xMAML model

An FPGA architecture allows to implement any kind of logical equation realized by logical blocks (CLB for Configurable Logic Bloc for Xilinx architectures). Each logic block contains some Look up tables (LUT) and flip-flops. We designed an embedded FPGA based on the CLB of the Xilinx XC4000 architecture [17] for its simplicity. This architecture was chosen to obtain realistic results on the WCDMA application by the use of the Xilinx synthesis flow. As we intend to use the dynamic reconfiguration, we synthesized each block of the WCDMA application on the XC4000 architecture in order to determine the number of logic cells required. The biggest function (Searcher) requires a maximum of $N_{PE} = 1235$ logic cells to be implemented. This result indicates that an array of 1235 logic cells is sufficient to execute the WCDMA if we can achieve the temporal constraints of $t_{aRt} = 22.2\ \mu s = \frac{t_{slot}}{3}$. Unfortunately, as we have to switch from one function to another, we need to implement preemption mechanisms leading to an available time of $t_{aRt} = 11.1 \mu s$ for bitstream propagation.

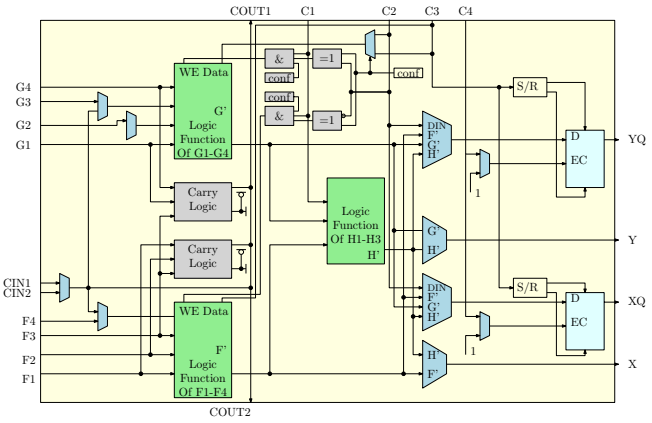

Figure 8. Logic Cell of the eFPGA based on the XC4000 architecture

The logical blocks of the studied eFPGA(Fig. 8) are composed of two 4-input LUTs and one 3-input LUT. The LUTs can be used together or separately. Two carry-chain functions are also included in the logical block. Each block contains two flip-flops which can be used or bypassed. The configuration of each cell is achieved by several multiplexers and requires $N_{bconfPE} = 66$ reconfiguration bits. Within an FPGA, the connections are made at the bit-level. The interconnection blocks are then supposed to have 24 outputs and 16 inputs. Considering that the interconnections could connect each input to the 24 possible outputs, 4 bits are necessary to configure each output. The number of inputs and outputs is parts of the exploration parameters, but are fixed here for sake of clarity. The FPGA is an array of 1235 logic cells interconnected by 1235 interconnection blocks. Once the xMAML description is realized, it is possible to explore the dynamic reconfiguration parameters.

The analysis of the xMAML description indicates that an interconnection resource needs $N_{bconfInterco} = 96$ bits for its configuration. Therefore, $B_s = 1235 \times (66 + 96) = 200070$ bits are required for each configuration context (FIR filter, Rake, Searcher). The configuration memory should be

able to store 600 210 configuration bits. A typical SRAM memory, as the one which could be used to save the contexts, is able to read/write at a frequency of 300MHz. Therefore, according to equation 3, $\frac{200070}{CBW} \times (300.10^6)^{-1}$ seconds are needed to propagate the whole configuration context. nD the number of required domains can be evaluated according to equation 4

$$nD = \frac{(300 \times 10^6)^{-1} \cdot \frac{200070}{CBW}}{11.1 \times 10^{-6}} \approx \frac{60}{CBW} \quad (5)$$

Therefore, for a better area optimization, the configuration bus (CBW) should be a multiple of 4. An 8-bit width configuration path seems to be a good trade-off. Consequently, for the WCDMA implementation on the eFPGA, $nD = 8$ reconfiguration domains are needed.

The second exploration phase is achieved on the interconnection resources produced. Thanks to the graphs of Fig. 6 and Fig. 7, it is possible to estimate the resource costs. Thus, the chosen parameters leads to a power dissipation of $95\ mW$ with an area of $0.85\ mm^2$ for the interconnection. Compared to static resources, this corresponds to 1.45 power overhead and 1.67 silicon area overhead for one resource.

C. Exploration of the implementation on a xMAML model of a Reconfigurable Processor

The dynamically reconfigurable processor DART is a coarse-grain reconfigurable architecture developed mainly for 3G mobile telecommunication application domain.

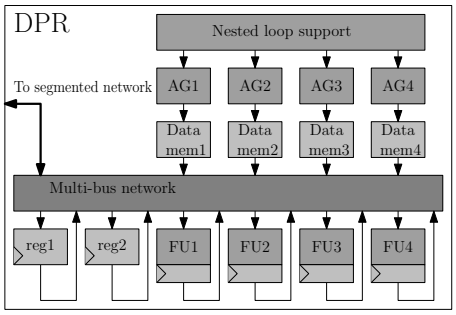

Figure 9. a DPR computing element of the DART reconfigurable architecture

DART architecture is build around computing elements called DPR (Fig. 9). Each DPR is composed of two registers, four address decoders to access four local memories, and four functional units (two adder/subtractor and two multipliers). The DPR is fully configurable thanks to a fully connected multi-bus. The original architecture was fixed and it was not possible to modify the structure of the DPR. Using an xMAML specification, it is now possible to specify all resources in a very flexible way and to explore various architectures. A DPR reconfiguration is executed in either 3 or 9 clock cycles. Each DPR requires $N_{bconfPE} = 38$ configuration bits. Therefore, 228 bits are needed for each configuration of the whole DART architecture. For each DPR, an interconnection resource supports 7 outputs and 11 inputs. Therefore, $N_{bconfInterco} = 28$ configuration

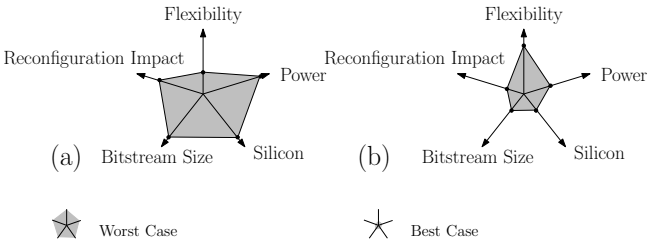

Figure 10. Exploration results for an implementation on eFPGA (a) and DART (b)

bits are required for each interconnection. This leads to a bitstream size of $B_s = 396$ configuration bits for the whole architecture.

The WCDMA implementation on DART uses 6 DPR. On the DART architecture, the reconfiguration is executed at the same frequency as the processing frequency, which is 130 MHz for the reference design.

Considering that no preemption is required on this architecture, the number of domains is:

$$nD = \frac{(130 \times 10^6)^{-1} \cdot \frac{396}{CBW}}{22.2 \times 10^{-6}} \approx \frac{0.14}{CBW} \quad (6)$$

Consequently, for a WCDMA implementation on DART, only one reconfiguration domain is needed. CBW can be tuned to the data size of the configuration memory, which should be able to store at least 1188 configuration bits. The graphs of Fig. 6 and Fig. 7 give an estimation of $96\ \mu W$ for the power consumption of the interconnect, with a total area of $0.024\ mm^2$. The overheads for the power and area are respectively 1.625 and 1.825 compared to static resources.

Fig. 10 shows the comparison between the exploration on the two target architectures (Fig. 10 (a) for the eFPGA and Fig. 10 (b) for DART). The figure gives a qualitative estimation of the performances of those architectures in terms of power consumption, silicon area, bitstream size, impact of dynamic reconfiguration and flexibility. On this picture, the smaller the gray area, the better for the corresponding parameter except for the flexibility parameter. First, the exploration on the flexibility is evaluated as the ratio between the number of reconfiguration resources and the bitstream size. As expected, the FPGA shows a good capacity to adapt its computing resource to a large spectrum of applications (thanks to the bit-level reconfiguration), which is not the fact for the DART architecture. Power consumption and silicon usage are estimated as the amount of resources needed for the implementation of the algorithm and required to support dynamic reconfiguration. From this point of view, DART has an advantage because of the algorithm chosen as proof-of-concept. These results may vary according to synthesis results and application target. The bitstream size is directly extracted from the xMAML analysis. This allows to estimate exactly the configuration data size and the size of the required configuration memory. The coarse granularity of DART permits to reduce these parameters and requires a smaller bitstream size. Finally, the reconfiguration impact

on silicon area is estimated from the bitstream size and the configuration register number. Due to the small data bitwidth used on FPGAs, dynamic reconfiguration applied on this type of architecture has a higher impact than in the case of DART.

Both explored architectures show advantages and drawbacks. The exploration has allowed to specify these particularities. Generally an FPGA implementation allows to keep flexibility high, but requires more silicon area and more configuration memory than a DART implementation. The results on area and power are better for DART which has been developed for the 3G telecommunication domain. We can see here that the exploration is thoroughly dependent on the application domain, and can be significantly simplified by the use of the xMAML description that needs to be done only once.

V. Conclusions

This paper has introduced a new architecture description language for the design and the modeling of dynamically reconfigurable architectures. Based on the MAML language, originally developed for the design of massively parallel processors, specific concepts for the description of dynamically reconfigurable architectures have been added to the original language. Firstly, the interconnection network description is more flexible, which is necessary to connect any kind of computing resources into a unique system. Secondly, configuration domains were introduced to split the architecture into several configurable areas. This concept allows to model partial reconfiguration, and to introduce mechanisms to support the implementation of preemption mechanism.

The efficiency of the methodology has been verified by the exploration of a WCDMA application on two xMAML architecture models. Models of an embedded FPGA and a dynamically reconfigurable processor were written and some parameters of the reconfiguration process (reconfiguration resources, reconfiguration area overhead, reconfiguration time, estimation of preemption cost, management of the reconfiguration) have been explored.

Future works will concentrate on the study of flexible compilation tools that can target the xMAML architecture model.

VI. Acknowledgments

This work has been performed in the context of the CoMap project and is funded by the French Ministry of Foreign Affairs. The authors would like to thank A. Kupriyanov, D. Kiessler, F. Hanning and J. Teich from University of Erlangen-Nüremberg for their fruitful collaboration.

References

[1] A. Kupriyanov, F. Hannig, K. Dmitrij, J. Teich, J. Lallet, S. Pillement, and O. Sentieys. Modeling of Interconnection Networks in Massively Parallel Processor Architectures. In *20th International Conference on Architecture of Computing Systems (ARCS)*, pages 268–282, 2007.

[2] H. Amano. A survey on dynamically reconfigurable processors. *IEICE TRANSACTIONS on Communications*, E89-B(12):3179 – 3187, Dec. 2006.

[3] S. Bashford, U. Bieker, B. Harking, R. Leupers, P. Marwedel, A. Neumann, and D. Voggenauer. The mimola language - version 4.1, 1994.

[4] V. Betz, J. Rose, and A. Marquardt. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, 1999.

[5] F. Charot and V. Messe. A flexible code generation framework for the design of application specific programmable processors. In *International workshop on Hardware/Software Codesign*, pages 27–32, 1999.

[6] A. Fauth, J. V. V. Praet, and M. Freericks. Describing instruction set processors using nml. In *European conference on Design and Test*, 1995.

[7] G. Hadjiyiannis, S. Hanono, and S. Devadas. Isdl: an instruction set description language for retargetability. In *34th annual conference on Design automation*, pages 299–302, 1997.

[8] M. Hariyama, W. Chong, S. Ogata, and M. Kameyama. Novel Switch Block Architecture Using Non-Volatile Functional Pass-Gate for Multi-Context FPGAs. In *IEEE Computer Society Annual Symposium On VLSI (ISVLSI)*, pages 46–50, 2005.

[9] A. Hoffmann, H. Meyr, and R. Leupers. *Architecture Exploration for Embedded Processors with Lisa*. Kluwer Academic Publishers, Norwell, 2002.

[10] O. Karatsu. Udl/i standardization effort another approach to hdl standard. In *Euro ASIC*, pages 388–393, 1991.

[11] D. Kawakami, Y. Shibata, and H. Amano. A prototype chip of multicontext FPGA with DRAM for Virtual Hardware. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 17–18, 2001.

[12] D. Koch, A. Ahmadinia, C. Bobda, H. Kalte, and J. Teich. FPGA Architecture Extensions for Preemptive Multitasking and Hardware Defragmentation. In *IEEE Conference on Field-Programmable Technology (FPT)*, pages 433–436, 2004.

[13] L. Lagadec and B. Pottier. Object-Oriented Meta Tools for Reconfigurable Architectures. In *Reconfigurable Technology: FPGAs for Computing and Applications II*, volume 4212, pages 69 – 79. SPIE, 2000.

[14] J. Lallet, S. Pillement, and O. Sentieys. Efficient dynamic reconfiguration for multi-context embedded fpga. In *Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2008.

[15] T. Ojanpera and R. Prasad. *Wideband CDMA For Third Generation Mobile Communication*. Artech House Publishers, 1998.

[16] S. Pillement, R. David, and O. Sentieys. Dart : A functional-level reconfigurable architecture for high energy efficiency. *EURASIP Journal on Embedded Systems*, 2008:13, 2007.

[17] Xilinx. *XC4000E and XC4000X Series Field Programmable Gate Arrays*, May 1999.