A Dynamically Reconfigurable Architecture for Low-Power Multimedia Terminals

Raphaël David, Daniel Chillet, Sébastien Pillement, Olivier Sentieys
ENSSAT - LASTI - University of Rennes 1, 6 rue de Kérampont, 22300 - France,
name@enssat.fr

Abstract: Within the framework of third generation telecommunication domain, reconfigurable architectures are becoming more and more popular thanks to both their flexibility and performances. In order to define a system that combines high-performance and low-energy consumption, a dynamically reconfigurable architecture, designed with energy awareness is proposed. This paper presents the main features of the DART architecture along with results from the application domain implementations. These results validate the architectural choices and demonstrate the adequacy between DART and next generation telecommunication applications.

Key words: Reconfigurable Architecture, High-Performance, Low-Power, UMTS

1. INTRODUCTION

In addition to the high performance requirements (more than 12 GOPS) inherent to multimedia processings or to access techniques such as W-CDMA (Wide-band Code Division Multiple Access) and to the low-energy constraints associated to portable devices (less than 500 mW), a third generation mobile communication system (Figure 1) brings new constraints to the semiconductor design world. In particular, the success of the Universal Mobile Telecommunication System (UMTS) will be linked to a greater flexibility of the standard than that of the Global System for Mobile Communication (GSM) or Interim Standard (IS)-95.

In fact, since UMTS integrates all the current generation networks, different types of processing will have to be supported by multimedia terminals. For example, speech signals has to be coded according to the

GSM norm with an Enhanced Full Rate (EFR) coder but also with a more powerful and adaptive AMR (Adaptive Multi Rate) coder which is recommended for the UMTS. At the same time, a multimedia terminal will have to support the evolution of different standards and services, mentioned in Figure 1, and the integration of new services which still have to be imagined.

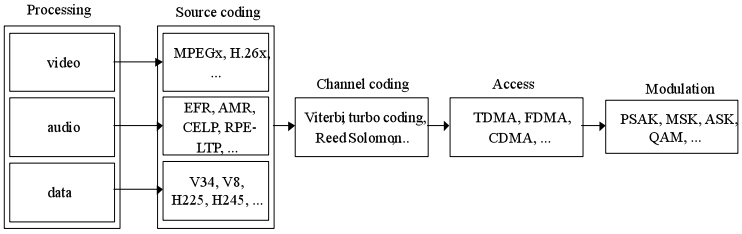

Figure 1. Block diagram of a third-generation transmission system

From an architectural point of view, furthermore this kind of flexibility which is usually referred to as software, the third-generation (3G) systems must have another kind of flexibility which is far more problematic: That of the Hardware. In fact, multimedia terminals will have to ensure successively the execution of very different applications in terms of calculation and data access patterns. For example, a Viterbi coder working at the bit level could follow an MPEG-2 coder working on 8-bit data. These processing modifications are then much more problematic since it will be necessary, in order to be efficient, to dynamically adapt the architecture to these changes.

Because of the lack of flexibility in ASICs and the low level of performance associated with the high energy consumption of the DSPs, the reconfigurable architectures are more and more taken into consideration to answer the problems associated with the 3G-telecommunications [11]. However, in spite of the quantity of projects on reconfigurable computing [8], none of them ambition to solve all the problems listed above. Some of these architectures are dynamically reconfigurable, others are multi-granularity or low-power but none of them is really adapted to this application domain.

Among these projects, some architectures related to our work can however be distinguished. The Pleïades project for example [1] is an architecture template supporting various granularity of calculation. Although this architecture has been designed under low-power constraints, it does however not met all our requirements. In fact, even if this architecture associates low-energy and high-performance, its flexibility is limited since it is domain specific. The dynamic reconfiguration proposed by the Chameleon^{TM} processor [13] allows a flexibility which does not limit its use to dedicated processings. This architecture, thanks to the large amount of

operators that are integrated, supports the 3G-telecommunication complexity but in spite of its flexibility and its performances, it is unusable in an embedded system because of its high-energy consumption.

Moreover these two examples, many other projects on reconfigurable computing are based on the use of FPGA. Some of these projects, like GARP [9] or NAPA [12], associate a reconfigurable circuit to a programmable processor or controller. Furthermore, other architectures such as Piperench [6] or RAPID [4] can be reconfigured at a higher level in a more efficient way, respectively at the operator and at the functional level. However, they do not meet all the requirements that come from the application domain. In fact, some are low power, others are very flexible or very powerful, but none of them associates the performance to the flexibility and the energy efficiency needed by portable multimedia terminals.

In order to answer to the overall problem of 3G telecommunications, a new architecture, called DART, is proposed. The aim of this paper is to present this architecture and to demonstrate its potential via some implementation examples. The next section focuses on the DART architecture. In the section 3, on the basis of key component study, we will estimate the level of performance and the energy efficiency of DART as well as its adequacy with next generation telecommunication domain. This paper will finally evoke the software tools associated to DART in the conclusion.

2. THE DART ARCHITECTURE

So that the user can use only one development platform and do not have to worry about the interfacing of the architecture with the rest of the system, DART is fully autonomous. This architecture has been designed to get the programming model as simple as possible. This is obtained by organizing the architecture into a hierarchy which allows the partitioning of the development flow. This hierarchy concerns the calculation, the storage, the interconnection and the control resources.

2.1 Processing primitives

The diversity of the calculation granularity levels in a 3G data processing sequence led us to integrate two kinds of operator in DART. For bit-level operations, we use an FPGA core, which allows a reconfiguration at the gate level. For arithmetic processings, we use some Reconfigurable DataPath (DPR), with a reconfiguration at the functional level. In order to optimize the architecture according to the application, we exploit the dynamic reconfiguration at both level of granularity.

The arithmetic processing primitives in DART are the DPRs fitted in Figure 2. They are organized around functional resources and memories, interconnected according to a very powerful communication network.

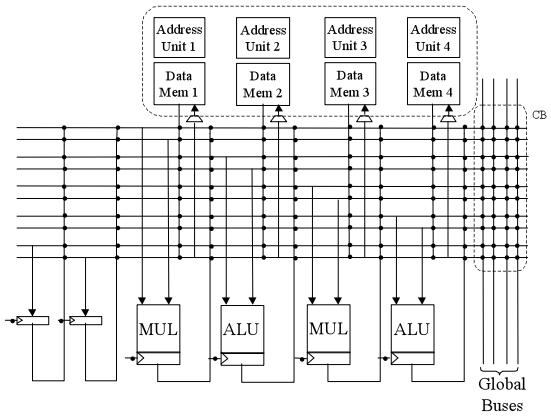

Figure 2. *Architecture of a DPR*

Every DPR have 4 functional units followed by a register, namely 2 multipliers (16x16->32) and 2 reconfigurable ALUs (32+40->40), supporting Sub-Word Processings. They are working on data stored in 4 local memories (256x16bits) which permit 4 read/write per cycle. In addition to these memories, 2 registers are also available in every DPR. These registers are particularly useful for data flow oriented applications where the different functional units are working on the same data flow but on samples delayed from one iteration to the following. In fact, in that case, these registers will be used to build delay chains that will allow the time-sharing of the data. All these resources are connected via an entirely connected network. The right part of the Figure 2 facts also of appearing some connections with global buses which permit to connect several DPRs for the massively parallel processings.

For the low-level processings, DART integrates an FPGA core. It will, for example, be very effective for the generation of Gold or Kasami code in W-CDMA [5], just as for channel coding algorithms. The operators are now LUTs, dynamically reconfigured as well as their interconnection network.

2.2 Cluster organisation

The processing primitives previously mentioned are integrated within the clusters of DART represented on figure 3. They integrate an FPGA core and 6 DPRs interconnected via a segmented network. Thus, every DPR can work independently from the others or be connected to them thanks to Switching Boxes that allow some flexibility in their connections. Every cluster also

integrates a data memory which is shared between all the processing elements, and a configuration memory dedicated to the FPGA. For its last, the reconfiguration will be done in a serial manner, thanks to the DMA controller.

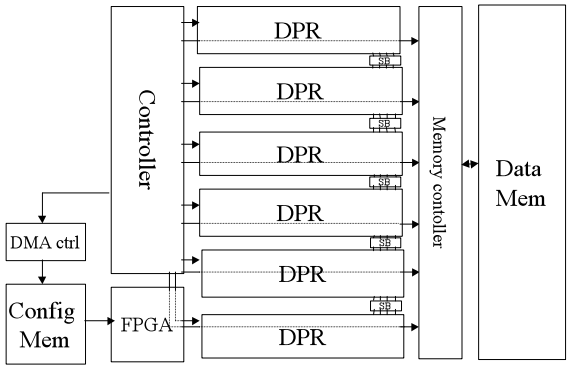

Figure 3. Architecture of a cluster

The cluster controller manages the DPRs and their reconfigurations. Its primary task is so to sequence the reconfiguration instructions of the cluster. Its architecture is similar to that of a controller in a typical DSP processor. However, it sequences configurations rather than instructions and so, it does not need to access an instruction memory at every cycle, but only when a reconfiguration occurs. This allows very significant energy savings.

2.3 Dynamic Reconfiguration

Resource	Bits/resource	Resource/DPR	Bits/DPR	Bits/cluster
Interconnect	92	1	92	592
Clock guard	1	10	10	60
Multiplier	3	2	6	36
ALU	11	2	22	132
total			**130**	**820**

Table 1. Reconfiguration instruction width

The DPRs are dynamically reconfigured due to instructions carried out from the cluster controller. The Table 1 resumes the targets of the reconfiguration and the number of bits necessary to this operation. This table shows that the dynamic reconfiguration of a cluster will require more than 800 bits. Wishing to be able to reconfigure the cluster in one cycle, to be effective in both regular and irregular applications, a disproportionately large amount of memory would have to be used. A more thorough examination of Table 1 led us to discern two kinds of processing.

2.3.1 Hardware reconfiguration

The regular processings are typically those that can be found in loop kernels. They are realized during long periods of time and are composed of very few operations. In order to optimize the datapath for the calculation pattern during these regular processings, a hardware reconfiguration will be done. The configuration being used the time of the processing, its modification is very occasional and it can so require a large amount of data without disturbing the execution of the entire processing. Moreover, the reconfiguration periods of the different DPRs in the cluster will be disjointed unless these DPRs are executing the same task.

The reconfiguration can so concern only one DPR per cycle without lowering the performances. This property allows the controller to manage only one instruction per cycle and so to be less complex while authorizing the simultaneous reconfiguration of several DPRs in the cluster. In that case, every DPR concerned by the reconfiguration will have its datapath optimized for the same processing pattern. We define this as the *Single Configuration Multiple Data* (SCMD) concept. This hardware reconfiguration will require between 4 and 9 instructions of 52-bit width.

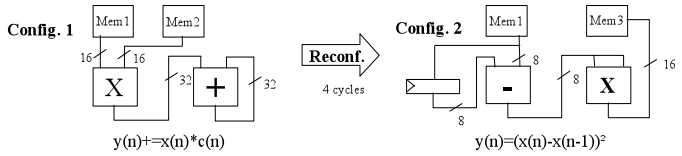

Figure 4. Hardware reconfiguration

This kind of configuration can for example be illustrated by the Figure 4. In this figure, the datapath is optimized at first in order to compute a filtering based on Multiply-ACcumulate operations. Once this configuration has been specified, the computation model is of dataflow type and no other instruction memory readings are done during the time of the filtering. At the end of the computation, after a reconfiguration step which needs 4 cycles, a new datapath is specified in order to be in adequacy with the computation of the square of the difference between x(n) and x(n-1). Once again, no control is necessary to conclude this processing.

2.3.2 Software reconfiguration

On the other hand, all the processings in UMTS are not regular and DART must be able to execute very different processings, from one cycle to the following. The efficiency of the processing is thus less important since it will be executed once. This property imposes to minimize the amount of data

necessary to the reconfiguration and therefore, the flexibility of the DPRs. It has been decided to adopt a calculation pattern of *Read-Modify-Write* type, such as those that are used in conventional DSPs.

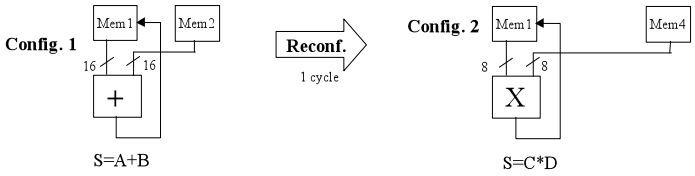

Figure 5. Software reconfiguration

This software reconfiguration thus concerns only the functionality of the operators and the origin of the data handled by the application. Thanks to these flexibility limitations, the DPR may be reconfigured at each cycle with only one 52-bit instruction, as illustrated in Figure 5. Like in hardware reconfiguration, the controller handles only one instruction per cycle which is sufficient since irregular processings have little parallelism.

2.4 The address generation units

Since the controller task is limited to the management of the reconfigurations, DART must integrate some dedicated resources for address generation. These units must provide the addresses of the data handled in the DPRs for each memory during the dataflow tasks. In order to be efficient in a large variety of application, they support numerous addressing pattern (bit-reverse, modulo, pre/post increment,…). These units are built around a module in charge of sequencing the accesses to an instruction memory (64x16-bits). In order to minimize the energy consumption, these accesses will take place only when an address has to be generated. For that, the sequencer may be put in Wait State thanks to an instruction which will moreover specify the number of Wait State. Another module is then in charge of waking up the sequencer after the number of cycle specified in the instruction. Even if this method needs some additional resources, its interest is largely justified with the energy savings.

Once, the instruction has been red, it is decoded in order to control a small datapath which will supply the addresses. On top of the four address generation units of each DPR (one per memory), a module will provide a zero-overhead loop support. Thanks to this module, up to four levels of nesting will be supported, each loop kernel being able to contain up to eight instructions, without any additional cycle for their management.

2.5 System level architecture of DART

As it has been said below, DART is fully autonomous. Hence, DART integrates a task controller which manage 4 clusters accessing a same data memory space. At this system view (Figure 6), the clusters can now be considered as the processing primitives of DART and the task controller is in charge of assigning the different tasks to be executed on the clusters according to urgency and resource availability constraints. It has to support a Real-Time Operating System. The configurations of the clusters are realized dynamically, since the task controller as only to specify to the cluster controller, which task has to be executed. The configuration support is thus only an address bound which corresponds to the location of the program in the cluster memory. In the same time, the data have obviously to be loaded into the cluster memories.

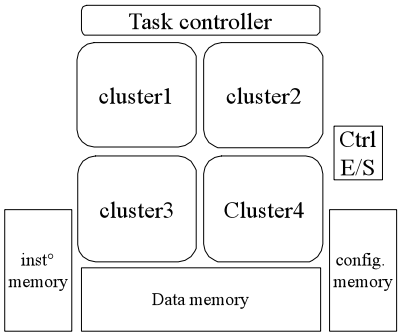

Figure 6. System level architecture of DART

3. DART EVALUATION

In order to validate the architecture and to evaluate its potential, some key applications of the UMTS have been implemented on DART. To estimate the performance and the energy efficiency of DART, its main parts have been synthesized on a 1.95V ST 0.18µm technology thanks to the Synopsys design tool framework. These synthesis have permitted to extract the key features of our design and to integrate them into the DART simulator which has been developed in systemC. The results described bellow are coming from this simulator.

3.1 Some key applications

Because of the complexity of W-CDMA [10], its implementation in a multimedia terminal has to be very effective. In order to test DART for such application, a subset of the norm has been implemented, a finger of a rake receiver which realizes the complex despreading, represented by the equations 1 and 2, with a spreading factor of 256.

$$I_{NB} = \sum_{j=0}^{SF-1} I_{WB} * C_P(j+\tau_i) + Q_{WB}(j) * C_Q(j+\tau_i) \quad (1)$$

$$Q_{NB} = \sum_{j=0}^{SF-1} I_{WB} * C_Q(j+\tau_i) + Q_{WB}(j) * C_p(j+\tau_i) \quad (2)$$

To illustrate the video processings we also have implemented a Discrete Cosine Transform, which is working on 8x8 pixels MacroBlocs, since this kind of algorithm is nearly systematic in video compression's standards like MPEGx or H.26x [7]. The 2 loop kernels of this algorithm are based on a *Multiplication-ACcumulation*. For MacroBloc Raws, the loop kernel is :

$$value[y] = \sum_{j=0}^{7} coef[y][x] * bloc[j + x * 8] \quad (3)$$

The multimedia terminals of third generation will still be used for the transmission of speech between two distant persons and so need very effective speech coders [2]. To illustrate these kind of processings, we have implemented an autocorrelation, on a signal of 240 samples, preamble of Levinson-Durbin algorithm which permits the adaptation of the prediction filter coefficients in speech coders such as the EFR or the AMR. The mathematical description of this processing is given by equation 4.

$$r(p) = \sum_{0}^{239} x(n) * x(n-p) \qquad for\ p \in [0..239] \quad (4)$$

3.2 Experimentation results

The Table 2 resumes the implementation results and reveals the potential of DART in two ways. The columns of this table specify for each application: the number of DPRs needed for the implementation; its number of operation; the number of execution cycles; the number of accesses to the instruction and to the data memories and finally the energy consumed for the execution of the algorithm.

Application	DPR	Operations	Cycles	Inst. Read	Data read	energy
Complex Despreading	2	2048	258	4	1032	435.8nJ
DCT 2-D	4	2048	85	6	1088	60.3 nJ
Autocorrelation	6	57600	2543	43	5040	3.15µJ

Table 2. Implementation results on DART

3.2.1 Performance analysis

A synthesis of the DPR estimate the operating frequency of DART at 130 MHz. Running at 130 MHz, DART is thus able to provide 260 MMACS (260.10^6 Multiplication-ACcumation per second) on each DPR when handling 16 bits data. Integrating 6 DPR per cluster, 1.56 GMACs/cluster can be achieved and these figures are doubled when handling 8-bits data as in video-coding [6]. From an instruction point of view, DART may deliver up to 520 MIPS/DPR or 3.12 GIPS/cluster. As an instruction includes an address generation, a memory access and up to 2 operations per multiplier (1 shift + 1 multiply) or 3 operations per ALU (2 shift + 1 ALU operation), this may be translated in 10.9 16-bit GOPS/cluster or 18.7 8-bit GOPS/cluster.

On each application previously quoted, the flexibility of the DPRs permits to obtain very good performances. The Table 2 shows for example that a finger of a rake receiver may be implemented on 2 DPRs which is allowing us to integrate 3 fingers in each cluster for a processing power bigger than 3.6 GOPS.

The connection of the DPRs being made thanks to a segmented mesh network, they can work independently or together. This property may be particularly useful. For example, to compute simultaneously a 2-D DCT and a finger of a Rake Receiver, within a same task, it would be possible to have the two elementary computations running concurrently on a single cluster. On the other hand, the massively parallel processings such as the autocorrelation should occupy all the DPRs, for a very effective execution.

3.2.2 Energy consumption analysis

If there are several architectures that can reach such level of performance e.g. the Chameleon [3], DART differs from its competitors by allowing a very significant energy saving thanks to the data sharing and the memory accesses minimization. In fact, the energy efficiency of ASIC is obtained by integrating operators that are not any larger or more complicated than they need to be and so, that consume a minimum of energy. Moreover, since they realize only one operation, the energy overhead of the instruction distribution is avoided. On the contrary, to be flexible, the programmable processors need general-purpose circuit blocks that are more complex to support the execution of several operations. Moreover, to control these general-purpose units, an instruction has to be red and decoded at each cycle. Since the instructions and the data are stored in very large memories, the energy waste in data access and control distribution widely exceed that which is useful to perform a computation. These two points are however, much less problematic in DART as shown in Table 2.

This table shows that only 43 instructions permit the control of the autocorrelation. On a conventional DSP processor more than 57,000 readings from the instruction memory would have to be done. Given the cost in energy of a memory access, the gain in energy consumption is therefore very important.

The second source of energy savings in DART is the data sharing. For example, on the autocorrelation, DART allows to divide by 12 the number of accesses to the data memory and so the energy waste due to this accesses (which is typically very high). This data sharing is made easier by the high degree of flexibility of the interconnection network and the use of the registers in data flow oriented applications.

In addition to the minimization of the memory accesses, the energy consumption of DART is lowered by the minimization of transistor activity. This point is essential in an architecture like DART since it must integrate, in order to be effective, a large amount of resources. However, all these resources will not be used at every cycle. DART has so to avoid making them to work when they are not in use to the execution thanks to guarded clocks. The energy is also saved in DART by optimizing the operators at the bit level and by scaling the voltage and the operating frequency of the clusters according to the complexity of the task to be implemented.

Hence, DART provide more than 9.2 MIPS for each mW consumed for 16-bit operations and about 15.8 MIPS for Sub-Word Processings. In an operation point of view, DART may deliver up to 32 MOPS for each mW consumed during the 16-bit operations and up to 55 MOPS for the Sub-Word Processings. It has to be noted that these 9.2 MIPS/mW are obtained in the worst, i.e. when at each cycle, each memory is accessed, 4 addresses are generated, and where the functional units realize an arithmetic operation on 16-bit data. This is however not the typical case since one of the main advantages of this architecture is to allow a large amount of data sharing. Practically, the energy consumption of the implementations described below are between 11.6 and 16.7 MIPS/mW.

4. CONCLUSION AND FUTURE WORKS

This paper has described a dynamically reconfigurable architecture for portable multimedia terminals developed within the framework of a project associating the University of Brest and ST Microelectronics, funded by the industry and research French ministry. We have shown, thanks to our implementation examples, that the use of a massively parallel architecture can be compatible with low-energy considerations. The next stage is to provide tool support for the development flow. This tool, which is in

development, is built around a retargetable compiler developed at the IRISA, CALIFE [14], able to supply the software configurations and on a behavioural synthesis framework, GAUT [15], developed at the laboratory which is in charge of generating the Hardware configurations. The codes provided by this tool can then be validate on athe DART systemC simulator.

References

1. A. Abnous and J. Rabaey. Ultra low-power specific multumedia processors. VLSI Signal Processing IX, pages 459-468, November 1996
2. T. Amada, K. Miseki and M. Akamine. CELP speech coding based on an adaptative pulse position codebook. In IEEE International Conference on Accoustics, Speech and Signal Processing (ICASSP), 1999
3. Chameleon Systems. Wireless Base Station Design Using Reconfigurable communications Processors. Technical report, 2000.
4. D. C. Cronquist, P. Franklin, C. Fisher, M. Figueroa, and C. Ebeling. Archi-tecture Design of Reconfigurable Pipelined Datapath. In Advance Research in VLSI, 1999.
5. E. Dinan and B. Jabbari. Spreading Codes for Direct Sequence CDMA and Wideband CDMACellular Network. IEEE Communications Magazine, 1998.
6. S. C. Goldstein, H. Schmit, M. Budiu, S. Cadambi, M. Moe, and R. R. Taylor. PipeRench: A Reconfigurable Architecture and Compiler. IEEE Computer, April 2000.
7. L. Hanzo and E.-L. Kuan P. Cherriman. Interactive cellular and cordless video telephony : State-of-the-art system design principles and expected performance. Proccedings of the IEEE, 2000.
8. R. Hartenstein. A Decade of Reconfigurable Computing : A Visionary retro-spective. In Design Automation and Test in Europe, 2001.
9. J. Hauser and J. Wawrzynek. GARP:A MIPS processor with a reconfigurable coprocessor. In IEEE Symposium on FPGA-based Custom Computing Machines (FCCM), June 1997.
10. T. Ojanpera and R. Prasard. Wideband CDMA For Third Generation Mobile communication. Hartek Publishers, 1998.
11. J. Rabaey. Reconfigurable Processing : The Solution To Low-Power Pro-grammable DSP. In ICASSP, April 1997.
12. C. Rupp, M. Landguth, T. Graverick, E. Gomersall, and H. Holt. The NAPA Adaptative Processing Architecture. In FCCM, April 1998.
13. X. Tang, M. Aalsma, and R. Jou. A compiler directed aproach to hiding configuration latency in chameleon processors. In International Conference on Field-Programmable Logic and Applications, April 2000.
14. F. Charrot and V. Messé. A Flexible Code Generation Framwork for the Design of Application Specfic Programmable Processors. In International Symposium on Hardware/Software Co-Design,1999.
15. O. Sentieys, J.P. Diguet and J.L. Philippe. GAUT : a High Level Synthesis Tool Dedicated to Real Time Signal Processing Application. EURODAC, September 1995.