DART : A Dynamically Reconfigurable Architecture Dealing with Future Mobile Telecommunications Constraints

Raphael David, Daniel Chillet, Sebastien Pillement, Olivier Sentieys
ENSSAT-LASTI - University of Rennes
6 rue de Kerampont, 22300 Lannion - France
name@enssat.fr

15 mai 2003

Résumé

In addition to the high performance requirements inherent to multimedia processings or to W-CDMA, future generation mobile telecommunications brings new constraints to the semiconductor design world. In fact, to support these processings, a system will have to be very flexible, in order to support the various algorithms allowed by the norm and the addition of new services, while keeping an energy consumption level compatible with the portability notion of this system. In order to associate high performances and low energy consumption in a flexible system, we developed a dynamically reconfigurable architecture called DART. The aim of this paper is to present this architecture and to estimate its level of performance and its adequacy with future generation mobile telecommunication systems.

1 Introduction

The original idea of futureeedscqsdcsde generation mobile telecommunications, illustrated by the Universal Mobile Telecommunication System (UMTS), is to integrate all current generation mobile networks associated to multimedia capabilities. In addition to the high performance requirements inherent to multimedia processings or to access technics such as the W-CDMA (Wide-band Code Division Multiple Access), a data processing sequence of third generation (Fig. 1) brings new constraints to the semiconductor design world. In fact, the success of the UMTS will be linked to a greater flexibility of the standard than that of the current generation mobile network such as the Global System for Mobile Communication (GSM) or the north American Interim-Standard(IS)-95. Hence, UMTS must be able to support various standards and algorithms for each kind of processing and to support their evolutions. For example, a speech signal can be coded according to the GSM norm with an Enhanced Full Rate (EFR) coding but also with a more powerful and adaptative AMR (Adaptative Multi Rate) coding, which is recommended for third generation telecommunications. Furthermore, UMTS will integrate new services as soon as they will be developed.

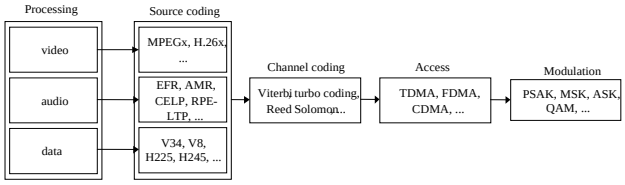

FIG. 1 – Block diagram of a third-generation transmission system

Moreover, from an architectural point of view, a multimedia terminal will successively have to ensure the execution of very different applications in terms of calculation and data access patterns, and which handles various data types. The lack of flexibility of ASIC and the low level of performance of DSP processors has led to study the alternative proposed by reconfigurable architectures

[16]. Nevertheless, in spite of the numerous project on reconfigurable architectures, none of them ambition to associate flexibility, high performance with the low power constraints inferred by the portability of these systems [12].

Among these projects, some architectures related to our work can however be distinguished. The Pleiades project for example [1] is an architecture template supporting various granularity of calculation. For example, MAIA [17], the specialization of this architecture for the speech coding associates some arithmetic operators like Multiplier-ACcumulators (MACs) and Arithmetic and Logic Units (ALUs) to FPGA cores. 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 consumption and high performance, its flexibility is limited since it is domain specific.

The dynamic reconfiguration proposed by the Chameleon^{TM} processor [20] allows a flexibility which does not limit the use of this architecture to dedicated applications. This architecture, thanks to the amount of operators that are integrated, allows the processing of third-generation telecommunication systems but in spite of this degree of flexibility and this level of performance, its high power consumption prohibits its use in an embedded system.

Moreover these two examples, many other projects on reconfigurable computing, are based on the use of FPGA. Some of these projects, like GARP [13] or NAPA [18], associate a reconfigurable circuit to a programmable processor who schedule the different tasks to be executed. Others architectures such as Piperench [9] or RAPID [4] can be reconfigured at a higher level in a most efficient way, respectively at the operator and at the functional level. However, they do not met 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 associate the high performance to the flexibility and the low energy consumption of the system.

In order to solve the overall problem of future generation mobile telecommunication systems, we develop a new architecture called DART. The aim of this paper is to present this architecture and the concepts used to associate high performance and flexibility in a low power architecture. In the first part of this paper the DART architecture will be described. Then, in the second part, we shall discuss about the performance and the energy efficiency of DART. We will moreover verify its adequacy with our application domain before to conclude on the work in progress and to come.

2 The DART architecture

Since a third generation telecommunication system should be able to handle several tasks concurrently, DART has been broken up into clusters, as illustrated in figure 2, that may work independently the one with the others. The first advantage of this decomposition is to limit the complexity of the controller which manages the circuit. Indeed, the primary function of this controller is to schedule the different tasks to be executed on DART' clusters according to their priority and resource availability. Thus, the controller do not have to sequence the instructions of every task but simply to specify to the clusters which task they have to execute. Hence, the control is distributed between the clusters which have their own controller to manage the processings of the task.

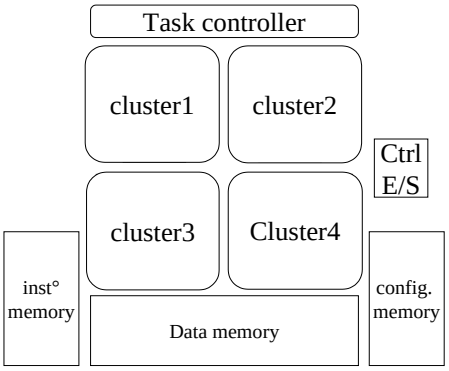

FIG. 2 – System level architecture of DART

The hierarchical organization of DART allows not only the distribution of the control but also that of the processing resources. Hence, it is possible to efficiently connect a very large number of resources without being too penalized by the interconnection cost. Indeed, distributing the processing resources allows the definition of hierarchical interconnect network that are much more efficient on a

performance and energy point of view, for large design, than typical global interconnect network [21]. With this kind of network, the lowest level of the resource hierarchy is fully connected, while the higher level communicates with segmented network. Since they are smaller and less loaded than that of a flat design, the local connections are much more efficient on a performance and energy point of view and thanks to the flexibility of this topology, the resulting architecture becomes a better target for the development tools.

The last but not least interest of this organization is to simplify the programming model of the architecture, by exploiting every parallelism level, i.e. thread, instruction and word levels. In fact, thanks to this decomposition, the development flow of an application may be partitioned into tasks liable to run concurrently. By breaking down the application as the architecture, it is then possible to obtain a development flow centered on the compilation of low complexity processings, e.g. loop kernel, on a very simple and orthogonal architecture presented in section 2.2.

2.1 Cluster architecture

As mentioned earlier, a mobile telecommunication system will successively have to ensure the execution of very different applications in terms of calculation or data access pattern and which handle different data types. In order to efficiently support the execution of both video coding and channel coding, which are respectively working at the arithmetic and at the logic level, each cluster of DART integrates two kind of processing primitives : some Reconfigurable DataPath (DPR in french) and an FPGA core. The DPRs, depicted in section 2.2, are reconfigurable at the functional level in order to optimize the interconnections between arithmetic operators (multiplier, ALU, ...) for the calculation pattern.

On the other hand, the FPGA core is reconfigurable at the bit level in order to efficiently support logic processings like the generation of Gold or Kasami code in W-CDMA [6]. Hence these two kinds of operators allow the adequacy between the algorithm and the architecture for a large set of applications like that of a data processing sequence of third generation.

Each cluster of DART, fitted in figure 3, integrates one FPGA core and six DPRs. The DPRs may be connec-

register and supporting SWP (sub-Word Parallelism) processings [7]. The use of the concept of SWP is justified by the numerous data sizes in a typical data processing sequence. In fact, even if audio and video coding are both arithmetic processings they are working on different data sizes (8 and 16 bits). Consequently we have developed some arithmetic operators that are optimized for the most common data format (16 bits) but which support SWP processings for the smaller data.

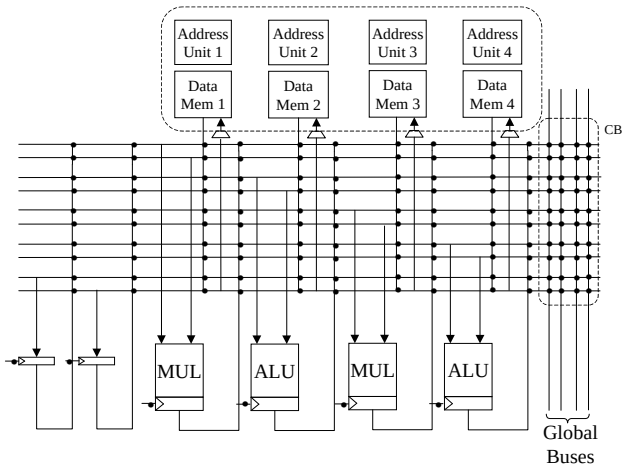

FIG. 4 – Architecture of a DPR

The functional units are dynamically reconfigurable (see section 2.3) and are working on data stored in 4 small local memories 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 that case, these registers will be used to build a delay chain to share the data in the time.

All these resources are connected via an entirely connected multibus network. The hierarchical organization of DART permits to kept these buses relatively small and so to limit its energy consumption. Thanks to this network, every resource may communicate with every others in the DPR and hence, the datapath may be optimized for every kind of calculation pattern. Moreover, this flexibility allows some data sharing, and so some energy

and are composed of very few operations, e.g. loop kernels. Those configurations being used for a long time, their modifications are very occasional and they can so require a large amount of data without disturbing the execution of the entire processing. Hence, in this case, a total flexibility of the DPR will be ensured and they will be reconfigured in 4 cycles, which is far less than the time of a typical regular processing. The datapath will thus be optimized according to the calculation pattern.

Moreover, we can notice that 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. This is defined as the *Single Configuration Multiple Data (SCMD)* concept. A hardware reconfiguration of the cluster will thus require between 4 and 9 instructions of 50-bits width.

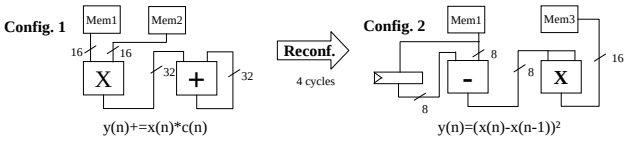

FIG. 5 – Hardware reconfiguration

This kind of configuration can for example be illustrated by the figure 5. 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 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 calculation of the square of the difference between x(n) and x(n -1). Once again, no control is necessary to conclude this computation.

2.3.2 Software reconfiguration

For irregular processings that need to change the configuration of the DPRs at each cycle, without particular order and in a non repetitive way, a software reconfiguration has also been defined. Thus, in order to be able to reconfigure the DPR in one cycle with an instruction of reasonable size, their flexibility has been limited. It has been decided to adopt a calculation pattern of *Read-Modify-Write* type, such as that of conventional DSP. In that case, for each operator useful for the execution, the data are red, computed, then the result is stored in the memory associated with this operator at each cycle.

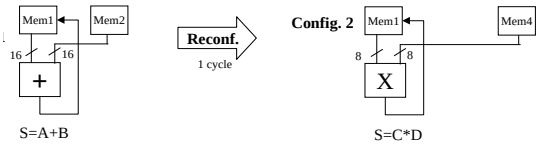

FIG. 6 – Software reconfiguration

This software reconfiguration thus concerns only the functionality of the operators, the size of the data and their origin. Thanks to these flexibility limitations, the DPR may be reconfigured at each cycle with only one 50-bit instruction. This is illustrated on figure 6 which represents the reconfiguration needed to replace an addition of data stored in the memories 1 and 2 by a SWP multiplication on data stored in memories 1 and 4. As for hardware reconfiguration, the controller handles only one DPR configuration instruction per cycle. This sufficient since the irregular processings have little parallelism.

Thanks to these reconfiguration modes, DART is able to support every kind of processings while being able to be optimized for the critical and regular processings. These two kind of reconfiguration can moreover be mixed without any constraints and even if the calculation efficiency of DART is lowered for the irregular processings, this drawback is not a real problem since these computations have a low complexity and have very little parallelism. The use of only one DPR is thus sufficient to support them.

2.4 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, i.e. the regular 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 at an instruction memory (64x16-bits). In order to minimize the energy consumption, this 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 its number of wait state. Another module is then in charge of waking up the sequencer after the number of cycle specified in the instruction.

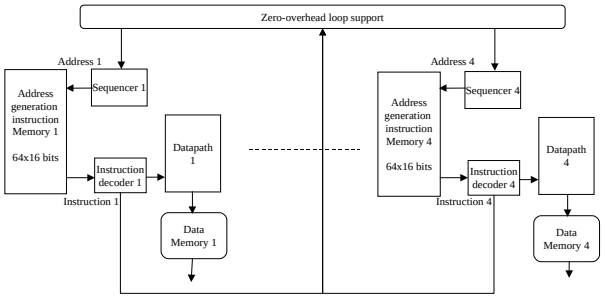

FIG. 7 – Address generation units and their zero-overhead loop support

Even if this method need 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 address. 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 its management. Two address generation units are represented on the figure 7 with their shared zero-overhead loop support.

3 Performance evaluation

This section analyzes the performance, area and energy consumption of DART. This analysis is based on some synthesis results of key components of the DPRs. These units, as well as their performances are described below, before the estimation of the performance of DART and of its adequacy with mobile telecommunications algorithms.

3.1 The functional units

The functional units have been synthesized with the *Synopsys* design tool framework with a 1.95V ST Microelectronics 0.18μm process. For the energy consumption estimation of the operators, done with *Design Power*, we have used 10,000 test vectors randomly generated. Hence, this estimation does not take into account the correlation between the data and is so slight pessimistic [5]. Thanks to the DART simulator, developed in SystemC [15], the overall energy consumption of this architecture is then evaluated from the activity of the different architectural modules (functional units, memory, glue-logic, …) and their energy consumption estimation realized with *Design Power*.

One of the critical elements in a DPR is the multiplier unit, at both energy, area and latency levels. This unit is constituted with a double precision signed 16-bit multiplier with saturation, followed by a shifter allowing the scaling of the data in the same cycle. The design of this SWP multiplier led to the conclusion that an important gain may be obtained by distinguishing the 8 and 16 bits multiplication and by integrating this two kind of operator. Even if this solution leads to a slight increase of the area (+10%) in comparison with a more classic 4-M decomposition of a 16-bit multiplier [3], it is largely compensated by the gain in latency (-30%). Moreover, as the 8-bit multipliers are not on the critical path of the unit, they can be optimized under energy consumption constraint. On the other hand, the 16-bit multiplier have to satisfy a trade-off between latency and energy consumption.

Finally, the unit is built around a 16-bit booth-Wallace multiplier and two 8-bit carry-save-array multipliers (Fig. 8). In order to minimize the energy consumption, some latches have been inserted in the design to stop the multipliers that are not in use during the execution. These

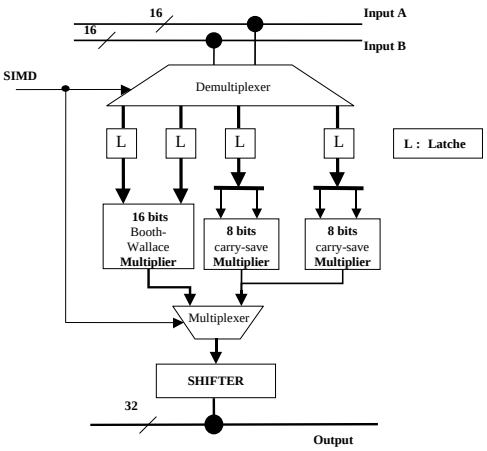

FIG. 8 – Multiplier Unit

latches led to an increase of the area (+5.8%) and of the latency (+11.4%) of the unit, but the energy consumption decreases appreciably (-23% for 16-bit multiplications and -72% for SWP 8-bit multiplications). Hence, the critical path of this unit is 4.21ns, the area is 40,443μm^2, and the energy consumed is of 129.7pJ for 16-bit multiplications and 47.6pJ for SWP 8-bit multiplications.

The second unit synthesized is in charge of the basic arithmetic processings such as addition, subtraction, comparison or logic processings (AND, OR, XOR, ...). In addition to these arithmetic and logic processings, this unit integrates a shifter to increase the precision of one of its input and another one for the scaling of the result. These scalings are realized during the same cycle than the arithmetic operation.

This unit is built around a 40-bit Schlantsky adder, support SWP processings for data coded on 8 or 16 bits, and allows the utilization of 8 guard bits to increase the dynamic of the data during the accumulation of operations like MAC. In order to minimize the energy consumed by the ALU, the different parts of the unit, i.e. shifters, logic unit, adder, ... handle data through a latch. Hence, the energy consumed by this unit for 40-bits operations is lower than 40pJ. The flexibility of this unit and its limited energy consumption come however at the price of a quite important latency since its delay is about 5.97ns and area since it needs 28,853μm^2 of silicon.

3.2 Energy efficiency and calculation power of a DART's cluster

A synthesis of the DPRs estimates the operating frequency at 130MHz. On the basis of this operating frequency estimation, DART will thus provide a calculation power on our ambition scale. Indeed, when handling 16-bits data, DART allows to realize 260 MMACS/DPR or 1.56 GMACS/cluster (this figure is doubled when the operators handle 8-bit data as in video coding). 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 and 1 multiply(+saturate)) 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.

Moreover this level of performance, DART must be efficient in an energy point of view. Thanks to the first consumption estimations, we can coarsely estimate the energy efficiency of DART. In fact, in a cluster, there are four major sources of energy consumption : the operators, the address generation units, the memory and the interconnection network. The address generation unit is built around an 8-bit ALU and one 64x16-bits memory and its energy consumption is 15.5pJ while a data memory access consumes 9.15pJ. Given those figures and those described in the previous paragraph, the energy efficiency of DART is about 9.2 MIPS/mW for 16-bits operations, and about 15.8 MIPS/mW for SWP operations. In an operation point of view, DART may deliver up to 32 MOPS for each mW consume during the 16-bits 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 case since the consumption is calculated for the case where at each cycle, each memory is accessed and so 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, there will not be 4 accesses to the memories at each cycle, since the operators will usually be chained, there will be a lot of 8-bit operations and these operations will not always be additions or multiplications but also scaling or logical operations. Practically, the energy consumption of the implementations described

below are between 11.6 MIPS/mw and 16.7 MIPS/mW.

3.3 Mapping an application on DART

In order to evaluate the adequacy between DART and mobile telecommunication algorithms some key applications of the UMTS have been implemented. These processings have been chosen because they are representative of three main kinds of critical processings in this application domain : the video processing, the speech processing and the W-CDMA.

To illustrate video processing we have implemented a Discrete Cosine Transform, which is working on 8x8 pixel macroblocs, since this kind of algorithm is nearly systematic in video compression standards like MPEGx or H.26x [11]. The 2 loop kernels of this algorithm are based on *Multiplication-ACcumulation*.

Even if the main critical processing of the W-CDMA is a FIR filter, we do not implement it since we already know that the regularity of this algorithm will be very well exploited on DART. Hence, to verify the effectiveness of DART for the W-CDMA [14], a subset of the norm has been implemented, a finger of a rake receiver which realizes the complex despreading with a spreading factor of 256.

Finally, since multimedia terminals of third generation will still be used for the transmission of speech between two distant persons and so need very efficient speech codec, 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 [2].

The table 1 resumes the implementation results. It 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 processing power of DART for the application ; the number of access to the instruction memory ; the number of access to data memories and finally the energy consumed for the execution of the algorithm.

On each application previously quoted, the flexibility of the DPRs permits to obtain very good performances. The table 1 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 treat simultaneously a 2-D DCT (4 DPRs) and a finger of a Rake Receiver (2 DPRs) it would be possible to have the two elementary tasks working concurrently on a single cluster, for a processing power of 4.9 GOPS. On the other hand, the massively parallel processing such as the autocorrelation should occupy all the DPRs, for a very effective execution.

If there are some architectures that can reach such level of performance, e.g. the Chameleon [19], DART differs from its competitors by allowing a very significant energy saving thanks to the data sharing and the memory access minimization. Indeed, the flexibility of programmable processors is usually obtained at the price of a large amount of energy waste in the data accesses and control distribution. In that kind of architecture, the instruction memory readings and the decoding of these instructions are responsible of most energy waste [10]. If this drawback is absent in FPGA type architectures, they suffer from the fine grain of their calculation resources. In fact, creating a 16-bits adder from logic cells will imply to make the signal to pass in transit through multiplexers, latches and other logic resources that are not useful for the execution. Thus, the latency and the energy efficiency of the resulting adder is lowered by these additional connections [8].

The ASIC solution, which does not suffer from these drawbacks, seems so to be the ideal one from a performance and energy efficiency point of view. However, as it has been said before, these architectures are unusable in our application domain since they are not flexible. Without pretending to reach the efficiency level of ASIC in performance or in energy consumption, DART can however be discussed as a new alternative to the traditional trade-off between flexibility and performance by integrating the dedicated operators described above and by minimizing the energy waste due to the control distribution through the design and to the memory accesses.

In fact, table 1 shows that only 5 instructions permit the control of the autocorrelation processing. Once these 5 instructions of configuration have been red, the only control that is necessary to conclude the execution is the sequence of the data addresses which are manipulated. On

Application	DPR	operations	cycles	GOPS	inst. read	data read	energy
Complex Despreading	2	2048	258	1.21	4	1032	435.8nJ
DCT 2-D	4	2048	72	3.69	5	256	64.7 nJ
Autocorrelation	6	57600	1520	2.97	5	3040	3.15 μJ

TAB. 1 – Implementation results on DART

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 these accesses (which is typically very high). This data sharing is made easier by the high degree of flexibility of the interconnection network and by the use of the registers in data flow oriented applications.

In addition to the minimization of memory accesses, the energy consumption of DART is lowered by the minimization of transistors activity. This point is essential in an architecture like DART since it must integrate a large amount of resources in order to be effective, even in the worst case. However, all the resources integrated will not be used at every time. DART has so to avoid making them 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 (see section 3.1) and by scaling the voltage and the operating frequency of the clusters according to the complexity of the tasks to be implemented.

4 Conclusion

In this paper, we have presented a dynamically reconfigurable architecture which supports the three main constraints of the third generation telecommunication domain : high performances, low energy consumption and flexibility. Thanks to our first synthesis results, we have shown that the dynamic reconfiguration can allow the elimination of a large amount of energy waste while offering a very high level of performances. The presentation of the implementation results of key tasks of the UMTS have moreover permitted to verify the interest of the use of functional reconfiguration which mainly targets the interconnection of arithmetic units. It has to be noticed that the results presented in this paper do not exploit the FPGA since the applications have only been implemented on the DPRs. Hence, to improve the performances of DART, the next step in our study is to develop an FPGA architecture which will allow the association of high performances and low energy consumption. Finally, since all the implementations presented in this paper have been done at the assembly level we will have to provide tool support for the development flow, in order to quickly provide a code optimized under performances and power consumption constraints.

Acknowledgments

This work is part of a project, associating the University of Brest and ST Microelectronics, funded by the industry and research French ministry. We would like to thanks B. Pottier (University of Brest), T. Ben-Ismael and O. Colavin (ST Microelectronics) for their contributions to this project.

Références

[1] A. Abnous and J. Rabaey. Ultra low-power specifi c multimedia processors. *VLSI Signal Processing IX*, pages 459–468, Nov. 1996.

[2] T. Amada, T. Amada, K. Miseki, and M. Akamine. CELP speech coding based on an adaptative pulse position codebook. In *ICASSP*, 1999.

[3] J. Cavanagh. *Digital Computer Arithmetic, design and implementation*. Mc Graw-Hill Computer Science Series, 1984.

[4] D. C. Cronquist, P. Franklin, C. Fisher, M. Figueroa, and C. Ebeling. Architecture Design of Reconfi gurable Pipelined Datapath. In *Advance Research in VLSI*, 1999.

[5] M. Denoual, D.Saille, J.-G. Cousin, and O. Sentieys. Fast Power Estimation at the architectural level. In *Design of Circuits and Integrated Systems Conference (DCIS)*, Nov. 2000.

[6] E. Dinan and B. Jabbari. Spreading Codes for Direct Sequence CDMA and Wideband CDMA Cellular Network. *IEEE Communications Magazine*, 1998.

[7] J. Fridman. Sub-Word Parallelism in Digital Signal Processing. *IEEE Signal Processing Magazine*, 17(2) :27–35, Mar. 2000.

[8] V. George. *Low Energy Field-Programmable Gate Array*. PhD thesis, University of California, Berkeley, 2000.

[9] S. C. Goldstein, H. Schmit, M. Budiu, S. Cadambi, M. Moe, and R. R. Taylor. PipeRench : A Reconfigurable Architecture and Compiler. *IEEE Computer*, Apr. 2000.

[10] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9) :1277–1284, sept. 1996.

[11] L. Hanzo and E.-L. K. P. Cherriman. Interactive cellular and cordless video telephony : State-of-the-art system design principles and expected performance. *Proccedings of the IEEE*, 2000.

[12] R. Hartenstein. A Decade of Reconfigurable Computing : A Visionary retrospective. In *Design Automation and Test in Europe (DATE)*, 2001.

[13] 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.

[14] T. Ojanperä and R. Prasard. *Wideband CDMA For Third Generation Mobile Communication*. Hartek Publishers, 1998.

[15] Open SystemC Initiative. SystemC version 2.0 Uers's Guide. Technical report, 2001.

[16] J. Rabaey. Reconfigurable Processing : The Solution To Low-Power Programmable DSP. In *IEEE International Conference on Accoustics, Speech and Signal Processing (ICASSP)*, Apr. 1997.

[17] J. Rabaey. A low-energy heterogeneous reconfigurable DSP IC. In *Design Automation Conference (DAC)*, June 2000.

[18] C. Rupp, M. Landguth, T. Graverick, E. Gomersall, and H. Holt. The NAPA Adaptative Processing Architecture. In *FCCM*, Apr. 1998.

[19] C. Systems. Wireless Base Station Design Using Reconfigurable Communications Processors. Technical report, 2000.

[20] 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*, Apr. 2000.

[21] H. Zhang, M. WAN, V. George, and J. Rabaey. Interconnect Architecture Exploration for Low-Energy Reconfigurable Single-Chip DSPs. In *IEEE Workshop on VLSI*, Apr. 1999.