Highly Scalable Dynamically Reconfigurable Systolic Ring-Architecture for DSP applications

Gilles Sassatelli, Lionel Torres, Pascal Benoit, Thierry Gil, Camille Diou, Gaston Cambon, Jérôme Galy
LIRMM, UMR UM2-CNRS C5506,
161 rue Ada, 34392 Montpellier Cedex 5, France
(33)04-67-41-85-69
{sassate torres pbenoit gil diou cambon galy}@lirmm.fr

Abstract

Microprocessors are today getting more and more inefficient for a growing range of applications. Its principles -The Von Neumann paradigm[3]*- based on the sequential execution of algorithms will no longer be able to cope with the kind of highly computing intensive applications of multimedia world.*
Nowadays approaches to deal with these limitations consist in the following:
- The first, and most natural way to increase the computing power is obviously to decrease the cycle execution time, thanks to new silicon technology: The functional frequencies for the newcomers CPUs are now getting on the way to 2 GHz.
- The second approach is co-design. The intended general purpose CPU will confide the computation of the most time demanding applications to a dedicated core. The most famous example are PC graphic cards which manage all the 2D and 3D display operations that even high-end CPUs are not able to handle efficiently.
Both methods are not satisfying. The first one quickly finds its limitations in however limited functional frequencies and power consumption reduction, as the second requires the design of a new core for each intended algorithm. New parallel execution based machine paradigms must be considered. Thanks to their high level of flexibility structurally programmable architectures are potentially interesting candidates to overcome classical CPUs limitations.
Based on a parallel execution model, we present in this paper a new dynamically reconfigurable architecture, dedicated to data oriented applications acceleration. Principles, realizations and comparative results will be exposed for some classical applications, targeted on different architectures.

1. Introduction

Nowadays silicon technologies allow the integration of entire systems on the same silicon die (SoC: System on Chip), putting together lots of different IP cores (Intellectual property). New highly constrained embedded products like cellular phones, pocket PCs are mostly based on a SoC approach (Figure 1). They are now facing new performances problems and need to be multimedia-ready for the incoming high bandwidth third generation networks like UMTS (up to 2Mbit/s bandwidth).

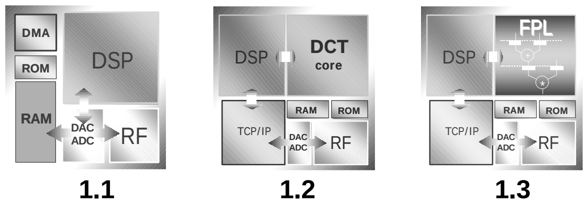

Figure 1. The three main SoC approaches

There are different ways to face these new problems:

- Use a more powerful or higher clocked DSP/µP (figure 1.1) than the ones used at present time; but it will probably not be feasible for the most demanding applications, as the resulting processor will grow until the size of a several hundreds megahertz Pentium-like core (such as the ones which take place in the most powerful PDA or pocket PCs), with the corresponding area, cost and consumption problems.

- Another way consists in the use of a dedicated core to compute the common parts of the algorithms in a given application field (Figure 1.2). For example, if JPEG and MPEG based applications are targeted, we will make the choice of implementing a wired IDCT (Inverse Discrete Cosine Transform) core, which is known to be the common most time consuming part of both algorithms[8]. An interesting, but restrictive solution as the application field is thus not extensible.

- Yet another way is the reconfigurable computing [2][3]. These kind of architectures are intended to exploit the parallelism of algorithms, by assigning multiple concurrent operations to multiples reconfigurable cell. FPGAs cores[1][11] are the most famous reconfigurable architectures, here depending on the target application different algorithm/architecture solutions could be synthesised (figure 1.3).

Application areas of such reconfigurable high-performance accelerators for wireless communication networks are potentially huge. The end-users portable device becomes field-adaptable by downloading software and configware (for the reconfigurable core) over the network down to the customers location: for instance for bug fixes, updates for changes (protocols and others), upgrading for new services (e. g. for innovating the network services), adaptation to environments, or, regional mobile phone standards (multi-standard handies!).

2. Reconfigurable solutions

Structurally programmable circuits like FPGAs offer a real alternative: providing new designs is replaced by reconfiguration within seconds. But one who choose FPGAs as accelerator[3] must pay the price for the ultimate flexibility, the fine-grained (bit-level) reconfigurable technology used shows its limitations in dataflow oriented applications where multiple arithmetic operators have to be synthesized. The highly combinational, regular structure of these ones are very area costly and inefficient (low functional frequencies) when mapped on FPGA technology.
A study at MIT reports[12], that FPGAs use only one percent chip area for the real application, whereas the other 99% are used for reconfigurability artefacts (about 10% configuration code memory, and about 90% for programmability of interconnect).
In our context of digital signal processing oriented applications (dataflow dominated) the fine grain architecture of FPGAs is not suitable, as we primarily target arithmetic level operators synthesis. Coarse grain reconfigurable blocks are the alternative, providing hardwired adders, multipliers, configurable at the word level, they are also much more efficient.
Our Systolic Ring architecture features a highly optimized, DSP-like coarse grain reconfigurable block (Figure 3) detailed on chapter 4.

3. System overview

Dataflow oriented applications require the use of coarse grained reconfigurable network. In this way, our approach follows an original concept (figure2):

- The operative layer is no longer CLB based, but use a coarse-grained granularity component: The Dnode (Data node). It is a datapath component, with an ALU and a few registers, as shown in figure 3. This component is configured by a microinstruction code.
- The configuration layer follows the same principle as FPGAs, it's a RAM which contains the configuration of all the components (Dnodes and interconnect) of the operative layer.
- We also use a custom RISC core with a dedicated instruction set as configuration controller; its task is to manage dynamically the configuration of the network and also to control the data communications between the reconfigurable core and the host CPU.

This architecture is thus not intended to be a stand-alone solution, rather an IP core accelerator for data oriented intensive computing, which would take place in a SoC. Figure 2 shows schematically our system in a SoC context. The µP can thus confide the most demanding part of a given application to our IP core.

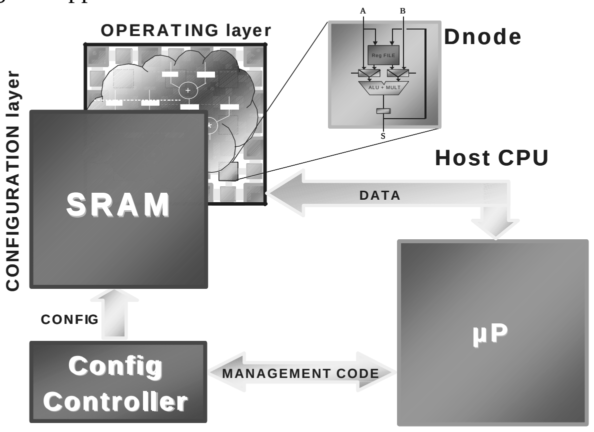

Figure 2. System overview

From a functional point of view:
- The operating system running on the host processor loads a given applications, specially designed for a co-execution. The application is constituted by host-executable code (directly loaded on the host memory) and Systolic Ring configuration controller executable code (management code).
- The host processor first uploads the management code to the configuration controller memory (which has it own program memory). This object code is specially designed to manage dynamically the configuration of the network (the content of the RAM thus changes from one cycle to

another), as to say, the functionality of the operating layer. Each clock cycle, the configuration controller is able to change up to the entire content of the RAM thanks to its dedicated instruction set.

- Once done, our core is ready to compute. The host processor sends the data to the operating layer via a specific scheme and then get back the computed data. As the configuration is dynamically managed, it is possible to multiplex the sent data, and to compute them by several sequential (hardware multiplexing) or concurrent(static) synthesized datapaths.

4. Operating layer architecture

4.1 Dnode architecture

This is the reconfigurable block of the system (figure 3), dedicated to word level computation, featuring a 16 bits arithmetic and logic unit (ALU) with an additional hardwired multiplier, a 4x16 bits register file and a special control unit.

Its architecture is optimized for digital signal processing applications. It is able to compute up to two arithmetic operations each clock cycle, as the adder and multiplier operators can be associated in a fully combinational way. Its instruction set features for instance a MAC operation using this resources, thus accelerating multiply-and-accumulate operations. All the possible operations can take place in a single clock cycle, even between two registers, with the result stored in one of these two registers (master-slave register architecture).

Its corresponding microinstruction code, the configuration code, comes from a memory location in the configuration layer. As previously said, this code evolves during the computing phase, the functionality can thus be changed from one clock cycle to another (from an addition to a multiplication for example).

Large realizations of dynamically reconfigurable architecture unfortunately imply non-linear network size / silicon area ratio, due to increasing routing needs and specially dynamical reconfiguration management.

For instance a 256 Dnodes version of our architecture, still fully dynamically reconfigurable (the entire network configuration can be changed within a few clock cycles) would requires a prohibitive, disproportioned RISC configuration controller.

Our way to solve this problem consists in a multi-level reconfigurable architecture; each Dnode has in fact two execution modes :

- Global mode (normal mode), already described : the Dnode executes the microinstruction code which comes from the configuration layer, managed by the RISC configuration controller.
- Local mode : the stand-alone mode : each Dnode has a special control unit constituted by 9 registers, a up to 8-states counter and a 8 to 1 multiplexer which forms a small local controller. Each one of the 8 first registers can contain a Dnode microinstruction code, and each clock cycle the counter increases the value on the multiplexer address input, thus sending the content of a register to the datapath part of the Dnode.

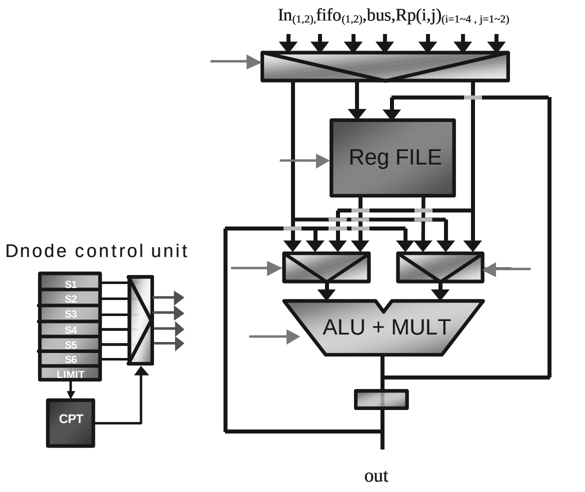
$In_{(1,2)}, fifo_{(1,2)}, bus, Rp(i,j)_{(i=1\sim4\ ,\ j=1\sim2)}$
Reg FILE
Dnode control unit
S1
S2
S3
S4
S5
S6
LIMIT
CPT
ALU + MULT
out

Figure 3. The Dnode architecture

In this last mode the Dnode is able to compute various algorithms like MAC, serial digital filters, FIFO emulation without RISC controller overheading. This scheme, joined to a specific input/output Data controller (exposed later) allows very efficient and high bandwidth data oriented computation.

4.2 The Ring architecture

Related works[2][3] propose mesh, array or crossbar-based operating layer architecture.

- *Mesh-based architectures*. Even very flexible, these ones usually suffer from routing problems. Each reconfigurable block must feature full routing capabilities with the nearest neighbours for direct communications. Routing over longer distances are achieved by dedicated lines, and with new silicon technologies allowing giant reconfigurable architectures, this requires important routing capabilities, with no-more maintainable propagation delays (general SoC problem, die-long interconnections cause hard timing problems).

- *Crossbar based arrays*. The routing capabilities are again usually quite satisfying, but area costly. The scalability of these architectures is also limited for the same reasons as mesh-based networks; and more specially FPGAs. The largest ones are facing propagation delay problems implying P&R tools to spend lot of time in routing phase.

- *Array-based architectures*. Aiming to map pipeline character of datapaths, they are often bi-dimensionals. Again, the feedback operations of all kinds of digital signal processing like algorithm require additional routing resources, area costly, and limits the scalability for next generations.

All these architectures share the same routing-relative problems, which will become prohibitive, and thus limit the scalability in next generations silicon technologies.
Our original approach to solve this problem takes place in the use of a curled bi-datapath structure:

4.2.1 Forward : The main Dataflow

We use a curled, pipelined systolic structure as shown in figure 4. All the D-nodes form a ring, which length (D-nodes layers number) and width (Dnodes per-layer number) can easily be scaled.
The Dnodes are organized in layers; a Dnodes layer is connected to the two adjacent ones by also dynamically reconfigurable switch components able to make any interconnection between two stages. It also manages data communications with the host processor by direct dedicated ports, and optional RISC communications via a shared bus. In normal mode, each Dnode can be seen as an arithmetic operator of a datapath which computes a data each clock cycle. In stand-alone mode each Dnode is like an autonomous CPU and can be seen as a 'macro operator'. The structure is also flexible in the way that all Dnodes have not to run in the same mode, allowing the Systolic Ring to compute either in global mode (normal mode), local mode (stand-alone) or hybrid (normal and stand-alone) mode.

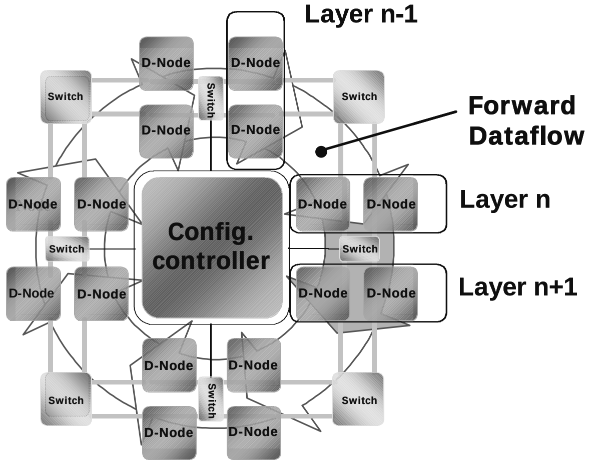

Figure 4. The Ring architecture

4.2.2 Reverse: The secondary Dataflow

The data feedback problem is addressed here: we use special feedback pipelines (figure 5), forming a reverse Dataflow to avoid complex routing structures. The last task performed by each switch is to write unconditionally (no control needed) the result computed by the previous Dnodes layer in a dedicated pipeline (each switch owns its pipeline), which allows the feedback of each data to the previous stages. Then these ones can choose to get these data through the switches, which have direct read access to all the pipelines. This technique ensures a good scalability of the architecture, as the routing problem is thus removed. These pipelines are also useful in RIF computation as the required delays on recursive branch are automatically achieved in them.

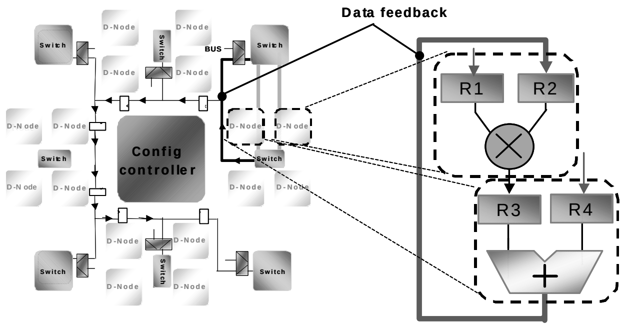

Figure 5. The Feedback network : An implementation example

5. Comparisons & realizations

5.1 Comparative Results

A 8 Dnodes, 16 bits wide data buses version has a maximal computing power of 1600 MIPS at the typical 200 MHz evaluated functional frequency, quite impressive compared to the 400 MIPS of a Pentium II 450 MHz processor. The theoretical maximum bandwidth of this version of the structure is about 3 Gbytes/s, limited to 250 Mbytes/s in our implemented communication protocol (a PCI based bus) between the host CPU and the core.
To program this structure we wrote an assembling tool, which parse both RISC level (for the control) and Ring level assembler primitives. It directly generates the machine object code, ready to be executed in the architecture.

5.1.1 Algorithm implementation

In the application field targeted by third generation systems we can find lots of video-relative techniques. Motion estimation (spatial redundancy removal, H.261 compliant) and wavelet transform (JPEG2000 compliant) are two well known computing intensive digital compression techniques. The exposed results in both implementations are achieved in a Ring-16, 16 Dnodes version of our architecture

Motion Estimation

Table 1 shows the performances of the Systolic Ring compared with the ASIC architecture implemented in [7] and Intel MMX instructions [8] using the criterion of the number of cycles needed for matching a 8x8 reference block against its search area of 8 pixels displacement.

Tab.1: Motion Estimation Performances

	Ring-16 @ 200 MHz	ASIC[7] @100MHz	MMX[8] @ 400MHz
Cycles	*3757*	*581*	*28900*

The ASIC implementation is much faster than our solution at the price of flexibility: the Systolic Ring provides the advantage of hardware reuse and is also almost 8 times faster than an MMX solution.

Wavelet Transform

Our implementation uses the lifting scheme algorithm [11] and operates a 2D direct transform on an 1024x768 pixels 16 bits coded image. One pixel sample is computed each clock cycle. Table 2 shows the comparative results of the differents wavelet transform implementations. The exposed results should take into account that 25 % of the Ring structure remains free for other computations.

Tab.2: Wavelet transform Performances

Circuit	Techno	Area	Frequency	Memory
[10]	0.7 µm	48.4 mm²	50 MHz	(768+30)x16
[11]	0.25 µm	2.2 mm²	150 MHz	897 Bytes
Ring-16	0.18µm	1.4mm²	200 MHz	N/A

Our structure shows again its efficiency in a such computing intensive context. All the exposed solutions are also able to compute a pixel sample per clock cycle, but area costly, and not flexible as dedicated to the wavelet transform.

5.2 Prototyping, Synthesis results & future work

The entire architecture (operating layer, configuration layer and configuration controller) has been described in VHDL. A 8 D-nodes, 16 bits data width version has been fully simulated, prototyped, and synthesised. The prototyping phase has been achieved on an Altera SOPC board featuring an APEX20K400 providing 400.000 equivalent system gates. Figure 6 shows the chose design implementation details. A Ring-8 (8 Dnodes) version including the configuration controller has been synthesized and implemented (RING-8). This core reads its configuration code from a preloaded memory (PRG RAM), and apply the corresponding computations on an 16 bits coded image also preloaded on another memory (IMAGE RAM). The resulting image is then wrote on video memory (VIDEO RAM) displayed on a monitor by an also synthesized VGA controller.

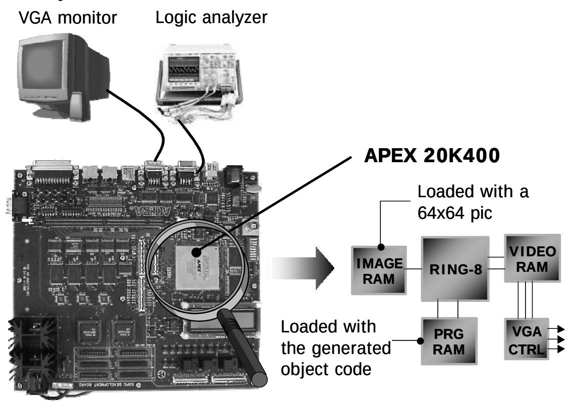

Figure 6. The APEX based prototype

Here are the synthesis results (table 3) in both 0.25µm and 0.18µm ST CMOS technologies. Areas and functional frequencies are Synopsys Design Compiler estimations.

Tab.3: Synthesis results

	0.25µm	0.18µm
D-node area	0.06 mm^2	0.04 mm^2
Core area	0.9 mm^2	0.7 mm^2
Est. Frequency	180 MHz	200 MHz

The low area of each D-node, joined to the exposed specific architecture shows that this one could easily be scaled to larger realizations. Figure 7 shows a foreseeable .18µm technology, 12 mm^2 die area SoC for high constrained embedded solution. Our specific architecture allows the integration of a powerful 64 Dnodes version of our Ring (3.4 mm^2 on-die area) with a widely used ARM7 CPU, able to run various operating systems like windows CE, Linux.

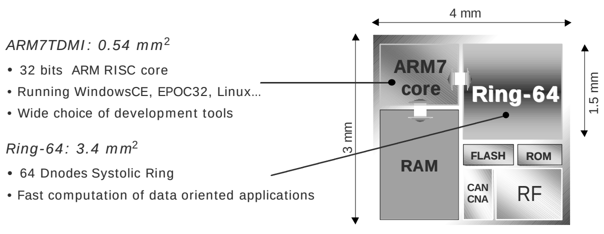
ARM7TDMI: 0.54 mm²
• 32 bits ARM RISC core
• Running WindowsCE, EPOC32, Linux...
• Wide choice of development tools
Ring-64: 3.4 mm²
• 64 Dnodes Systolic Ring
• Fast computation of data oriented applications
4 mm
3 mm
1.5 mm
ARM7 core
Ring-64
RAM
FLASH
ROM
CAN CNA
RF

Figure 7. A foreseeable SoC

This kind of solutions could provide a great computation power/cost trade-off, which combines the flexibility of a CPU / reconfigurable architecture couple with the efficiency of applications dedicated cores.

6. Conclusion

We have proposed a new coarse grained arithmetic block based dynamically reconfigurable architecture which proves its efficiency in data oriented processing. Its scalability shows that the application field can not only be limited to embedded high-constrained applications, but can also be very useful in other contexts, where data oriented high data bandwidth processing remains critical. A small 8-Dnodes version of this structure already provides up to 1600 MIPS of raw power for data dominated applications with a sustained data rate of 3 Gbytes/s at 200 MHz, either in global or local mode. Efficient dataflows control operations are usually hard to achieve in all kinds of coarse grain reconfigurable architectures due to the lack of fine grain logic (used to control FSM synthesis). This limits the field of forseeable applications to highly dataflow dominated algorithms, and for instance, the integration of a RIF filter using resource sharing (fewer operator than normally required on hardware inplementation) is impossible without very efficient dynamical reconfiguration implementation, allowing to change each clock cycle the functionnality (multiplication, and then addition...). Our approach featuring a dual layer configuration scheme (global and local) overcomes these limitations, allowing efficient concurrent Dnode configuration switching (control operations). This allows efficient algorithm ‘compilation’ by the ability to identify macro-operators (RIF, RII, FIFOs & LIFOs, trigonometric op., etc.) on the high level description, and directly map them onto Dnodes thanks to local mode.

Our future work takes place in the realization of an efficient compiling/profiling tool, the key to success of reconfigurable computing architectures.

7. References

[1] Stephen Brown and J. Rose, "Architecture of FPGAs and CPLDs: A Tutorial," IEEE Design and Test of Computers, Vol. 13, No. 2, pp. 42-57, 1996

[2] W. H. Mangione-Smith et al, "Seeking Solutions in Configurable Computing," IEEE Computer, pp. 38-43, December 1997

[3] R. Hartenstein, H. Grünbacher (Editors): The Roadmap to Reconfigurable computing Proc.FPL2000, Aug.27-30,2000;LNCS,Springer-Verlag2000

[4] C. Hsieh and T. Lin, " VLSI Architecture For Block-Matching Motion Estimation Algorithm," IEEE Trans. on Circuits and Systems for Video Technology, vol. 2, pp. 169-175, June 1992.

[5] ISO/IEC JTC1 CD 10918. Digital compression and coding of continuous-tone still images – part 1, requirements and guidelines, ISO, 1993 (JPEG)

[6] Xilinx, the Programmable Logic Data Book, 2000

[7].A.Bugeja and W. Yang, “A Re-configurable VLSI Coprocessing System for the Block Matching Algorithm”, IEEE Trans. On VLSI systems, vol. 5, September 1997.

[8] Intel Application Notes for Pentium MMX, http://developer.intel.com/

[9] Stéphane G. Mallat, “ A Theory for Multi-Resolution Signal Decomposition: The Wavelet Representation ”, IEEE Transactions On Pattern Analysis And Machine Intelligence, vol. 11, N 7, July 1989

[10] D. Navarro, “ VLSI Implementation of 2-D Mallat's Wavelet Transform ”

[11] C. Diou, L. Torres, M. Robert. « Wavelet Core for Video Processing ».In Proceedings ICIP’2000 Conference, Vancouver, Canada, September 2000.

[12] Why reconfigurable computing, Department of Computer Science, Computer Structures Group http://xputers.informatik.uni-kl.de/