ADAPTO: full-adder based reconfigurable architecture for bit level operations

G.C. Cardarilli, L. Di Nunzio, M. Re
Department of Electronic Eng.
University of Rome Tor Vergata,
Via del Politecnico1, 00133 Rome, Italy
Email: {marco.re, g.cardarilli}@ieee.org
Di.Nunzio@ing.uniroma2.it

Alberto Nannarelli
Department of Informatics & Math Modelling
Technical University of Denmark
Kongens Lyngby, Denmark
Email: an@imm.dtu.dk

Abstract— **Low cost microprocessors and DSPs are optimized to perform general arithmetic and logic operations on native wordlength. On the other hand, the efficiency decreases when they process shorter data (more clock cycles per operation are required). Recently different solutions have been proposed to overcome this problem. Among those, the one based on a main processor with a Reconfigurable Unit (RU) used as coprocessor (to speed up fine grained operations) is the most common. Typically those coprocessors, similar to FPGA, are composed by Look-Up Tables (LUTs) and pass transistors interconnects. In this way, due to the great number of reconfiguration bits, it is impossible to obtain together a run-time reconfiguration and an efficient implementation, avoiding idle hardware resources . This paper proposes a new dynamic reconfigurable architecture that can be embedded in microprocessors or low cost DSPs to accelerate the execution of the above mentioned operations. The goal of ADAPTO (Adder-based Dynamic Architecture for Processing Tailored Operators) is to reduce the hardware complexity and the reconfiguration time, with respect to typical LUT based Reconfigurable array. ADAPTO supports both hardware reconfiguration and instruction execution in the same processor clock cycle. This goal has been obtained by using a new reconfigurable unit based on full adders, instead LUTs, and simplifying the network interconnect.**

I. Introduction

The increasing performance of modern microprocessors makes possible to use them in a rising number of different applications. The main characteristics enabling this diffusion is the rapidly decreasing cost and the possibility to reuse developed architectures in a large number of applications. In order to improve microprocessor performances, a lot of efforts have been devoted to design more efficient instruction sets and powerful arithmetic/logic units [1], [2]. These architectures are conceived for portable and low-cost applications, and are used to implement both control and processing functionalities (as those required in multimedia, communication and image processing applications). Normally, standard microprocessor architectures are optimized to efficiently execute general-purpose instructions on data of the native wordlength. They normally need several clock cycles to execute instructions on shorter data[1], [2]. Unfortunately, the above cited applications frequently require to perform operations on variable data size. This is the case, for example, of pixel/char operations, cryptography loops, convolutional encoders. These *non-standard* operations heavily degrade the processor performance, precluding the real time execution of the chosen algorithm. In this paper a new RU based on full adders is proposed. In the following, we assume that the arithmetic unit of the microprocessor hosts all the arithmetic functions optimized for the native wordlength. The main requirements for of the proposed RU are:

- High flexibility (in order to be easily reconfigured for different operations, including the boolean ones).
- High reconfiguration speed, for resource reuse without the stalling of the processor during the program execution.
- Low cost in terms of silicon area and power

In the paper the main attention is devoted to the description of the computational part of the RU (the aspects related to the interconnect are only sketched) and its adaptability to different bit level processing applications is analyzed. In Section II the previous work is described, while in Section III the main characteristics of the ADAPTO architecture are discussed. Section IV presents some applications while in Section V some conclusions are drawn.

II. Previous work

Razdan and Smith [3], [4] (Prisc Architecture), and Hauck [5] (Chimaera Architecture) showed that it is possible to improve the execution speed for *non-standard* operations by inserting in the processor datapath a RU based on LUTs.

In this way, during the program execution, the processor performs the normal instructions, while the RU executes operations not directly supported by the microprocessor speeding-up the algorithm execution.

In the following, these special operations will be named Reconfigurable Operations (RO). By using this approach, the best execution speed is obtained by reconfiguring the RU as fast as possible. In this way the ROs are executed as soon as they are scheduled.

The main problem in using a RU based on LUTs is related to the impossibility to reload the configuration memory in a very short time. In fact, the use of LUTs implies the reloading of all the configuration bits, making it incompatible with a run-time reconfiguration. Similar problem arises for the reconfiguration of interconnect, but in this case it is possible to find simple solutions for reducing the bit amount. Some of these solutions are shown in sect. III-B. For this reason, every time we need to perform a RO, the following steps must be performed:

978-1-4244-1684-4/08/$25.00 ©2008 IEEE

1) Stopping of the program execution,
2) Loading of the configuration bits in the LUTs,
3) RO execution.

In order to increase the reconfiguration speed different strategies can be used:

1) Map several contexts in different parts of reconfigurable LUT array [5].
2) Use partial reconfiguration of LUTs in more complex structures (for instance the pipeline reconfiguration used by Piperench [6]).
3) Replace the LUTs with another element characterized by a simpler and faster reconfiguration procedure.

The main drawback of the first two solutions is the great hardware complexity and power consumption. Only a small part of the computational units (LUTs) are used during the execution of a RO. For example, in the second solution the partial reconfiguration does not allow the complete exploitation of the available resources (those involved in the reconfiguration phase).

Moreover, the reconfiguration policy could limit the algorithms that can be implemented. In particular, the architecture proposed in [6] is based on processing stripes and the pipeline reconfiguration is carried out on a subset of stripes. Consequently, during the RO computation this subset is not available for the processing. Moreover, only pipelined applications can be efficiently executed using this approach [7]. These characteristics reduce the performance and the flexibility of the structure, reducing the overall efficiency. For these reasons in this paper a new structure based on simpler computational element suitable for new microprocessors and DSPs architectures has been developed.

III. The ADAPTO architecture

In the following subsections the structure of the computational unit and the structure of the array are illustrated.

A. Computational Unit

In order to guarantee flexibility, high reconfiguration speed and low area overhead, the computational unit has been based on full adders instead of LUTs. The full adder can be easily configured for performing the following operations: one bit addition, NOT and PASS, 2 input AND, 2 input OR, 2 input XOR, 3 input XOR and 3 input XNOR. The different functions can be selected by forcing one or more input pins of the FA to a fixed value as shown in Fig. 1. For instance, a 2 input AND is obtained by putting the Cin input to 0 and taking as output the Cout pin. Clearly, this structure is less flexible than that based on a LUT, but on the other side its reconfiguration is faster because it requires to change fewer configuration bits.

Another advantage of this solution is the lower number of MOSFETs required for its implementation (for example [8] and [9] use respectively 14 and 10 MOSFETs). For ADAPTO we chose the adder proposed in [8] that assures an higher signal quality, introducing an output inverter.

The basic computational element of the new architecture, shown in Fig. 2, is the LB (Logic Block). It is based on two multiplexers, a selector (realized by a multiplexer with a suitable coding of selection bits) and a full adder. The input and the output multiplexer together with the selector are used for programming purposes. In particular inputs S0, S1, S2, and P are used to select the operation to be performed and the operands (Fig. 1). The signal C_O is directly connected to the signal C_{out} of the previous LB. If a zero carry is required (for example in the case of the LSB of a multibit adder) 0 input is selected by the configuration bits S_0 and S_1. The LB circuit uses about 50 transistors and its configuration requires only 4 bits. On the other hand, a similar structure based on a 3-inputs 2-outputs LUT needs about 48+26 = 74 transistors for circuit implementation and 18 bits for its configuration (16 bits for LUT and 2 for mux selection).

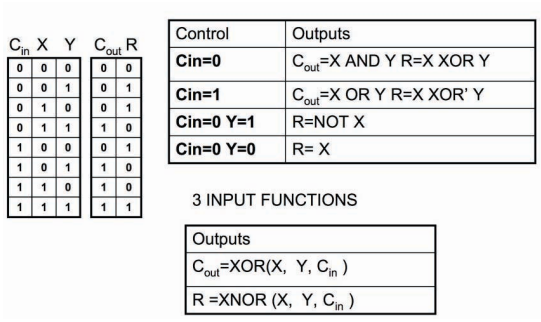

Fig. 1. Functions implemented by full-adder.

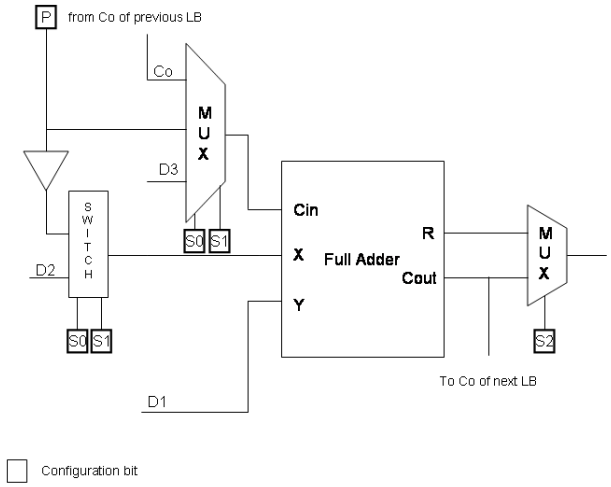

Fig. 2. Logic Block Architecture

B. Array structure

The ADAPTO architecture is based on alternated stripes of LBs and interconnects (see Fig. 3). In the proposed version three layers are used. The data flow is directed from the top to the bottom of the RU architecture. Each row of the array is composed by 32 LBs and each LB needs 4 configuration bits. Therefore with only 4*32 *3=384 bits it is possible reconfigure the whole computational unit (much less than the 1728 bits required by a LUT based implementation). The interconnection structure is shown in Fig. 4. This structure

is based on a multicontext approach (in order to allow high reconfiguration speed). Each LB output can be linked with any inputs of the LBs belonging to the stripe on the bottom row. In addition to the 32 inputs coming from the upper LBs, we have also added an additional line for interconnect configuration; depending on its value, shifting operations with 1 or 0 insertion can be implemented.

Interconnect is based on pass-transistor devices. Multicontext configuration bits are stored in local memories. Consequently, the control of interconnect reconfiguration requires only few lines for carrying the address of the multicontext memories. For N contexts, only $\log_2(N)$ addressing lines are required. Moreover, since only one of 33 transitors of the interconnect column is activated at each time (an input pin must be connected to a single signal source), it is possible to use a column decoder in order to reduce the size of the multicontext memory. In this way the number of configuration bits stored in the multicontext memories is decreased from $9504 * N$ (each row has 32 LB with 3 inputs that can be connected to 33 lines for each interconnection level) to $1728 * N$. The total amount of memory (program plus interconnect) is about 384*N +1728*N = 2112*N.

As described above, carry chain uses a direct interconnect (linking adjacent LBs) for the speeding-up of the carry propagation in multibit adders.

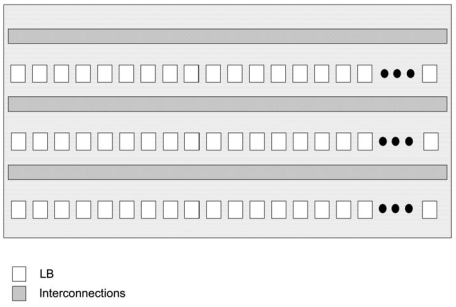

Fig. 3. Reconfigurable Array structure

IV. Applications

In order to verify the flexibility and the speed up factor obtainable by the ADAPTO architecture, different applications requiring different types of fine grained operations have been studied and implemented on the architecture.

A. Decoding

In the algorithms used in telecommunication applications there are often loops performing boolean operations on reduced wordlength data. These loops greatly reduce the performance of the algorithm implementations on RISC processors and DSPs. For example, consider the following pseudo-code extracted from the software implementation of the DRM receiver: [10]

```
A=A and (B<<C)
```

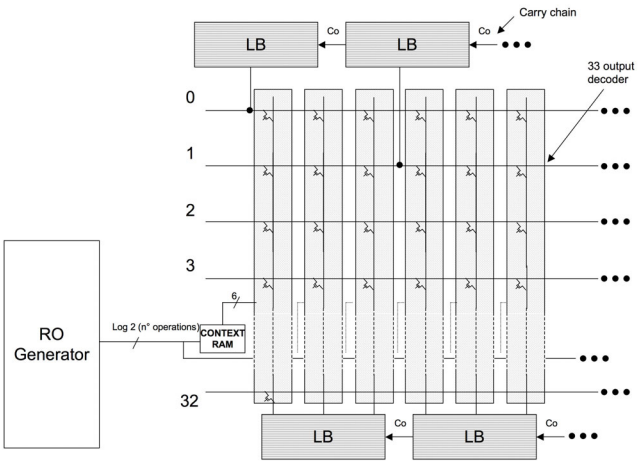

Fig. 4. Structure of the reconfigurable interconnect network.

To execute this line, a sequential ALU needs several cycles to perform the operation $B << C$, one cycle for the NOT, and one more cycle for the AND. Using the ADAPTO architecture configured as shown in Fig. 5 it is possible to execute the overall operation in one clock cycle.

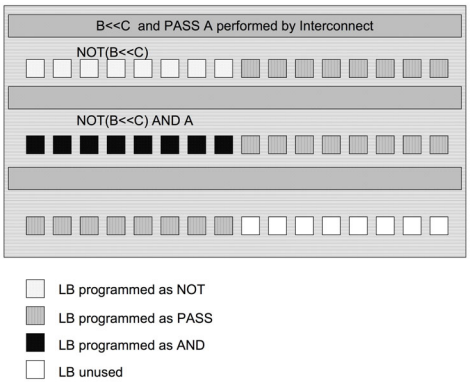

Fig. 5. ADAPTO implementation of a typical decoding operation

B. Pixel operations

Image processing algorithms usually perform operations on data with size shorter than the native processor wordlength. In the implementation of these algorithms with low cost DSPs, we have to face two problems related to the execution of arithmetic and boolean operations:

1) Arithmetic operations (add/sub operations): in this case the processor could waste time and HW resources because it cannot efficiently execute arithmetic operations on inputs shorter than its native wordlenght.
2) Boolean operations: usually microprocessors only execute one boolean instruction (on the whole word) in each cycle.

An example of an application that can be accelerated by using ADAPTO is the union of two monochromatic images. Let consider the two black circles shown in Fig. 6. The

algorithm performs the union of these two images in three steps:

1) Inversion of all the pixels of the two images using the NOT operator (Fig. 6 B)
2) logical OR between the resultant images (Fig. 6 C)
3) Inversion of the image produced in step 2 obtaining the final result (Fig. 6 D)

Using the ADAPTO architecture is possible to execute the algorithm in one clock cycle.

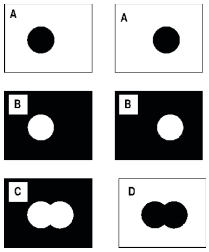

Fig. 6. Example of the union of two images

C. Convolutional coding

In this case the convolutional encoder used in Eureka147and DRM [8] standards, and shown in Fig.7 has been considered for the experiment.

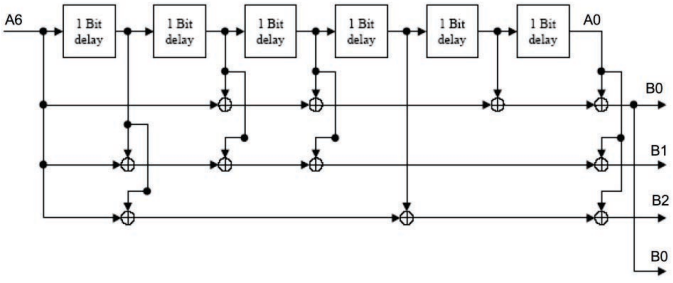

Fig. 7. DRM Eureka147 convolutional encoder

Using the ADAPTO architecture configured as shown in Fig. 8, the final result is obtained in one clock cycle.

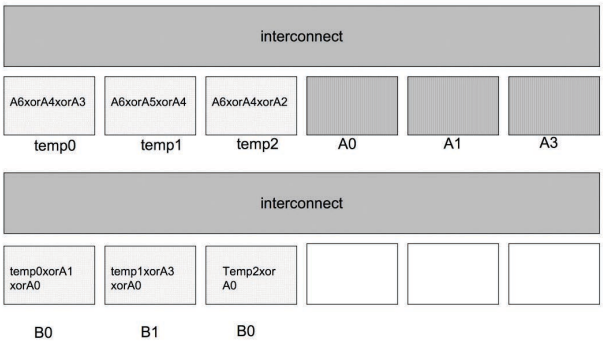

Fig. 8. Implementation on ADAPTO of a DRM/Eureka147 convolutional encoder(only six LB of only two row are shown)

In this figure, the outputs are named B0, B1 and B2 and the element stored in the shifter register (from the left to rigth) respectively A6, A5, A4, A3, A2, A1 and A0. The first row of the ADAPTO RU, is configured as 3 input XOR evaluating:

```
temp0 = A6 XOR A4 XOR A3
temp1 = A6 XOR A5 XOR A4
temp2 = A6 XOR A5 XOR A2
```

During this computation three full adders of the first stripe have been configured as pass gates in order to propagate to the second stripe the A0, A1, A3 inputs. The second stripe has been also configured as 3-input XOR functions performing:

```
B0 = temp0 XOR A1 XOR A0
B1 = temp1 XOR A3 XOR A0
B2 = temp2 XOR A0
```

In this way the one bit coding requires one clock cycle.

V. Conclusions

This paper proposes a new dynamic reconfigurable architecture that can be embedded in microprocessors or low cost DSPs to accelerate the execution of fine grained operations. The goal of the proposed architecture is to reduce the hardware complexity and the reconfiguration time with respect to those based on LUT. This objective has been achieved using full-adder based LBs. The solution allows the increasing of the hardware efficiency (reduction of the number of transistors) and the reduction of the reconfiguration bits. The presented architecture supports both hardware reconfiguration and instruction execution in one processor clock cycle. In order to check the effectivity the ADAPTO architecture, several applications have been verified.

Acknowledgment

The research work has been supported by the Otto Mønsteds Fond in the context of a Visiting Professor Sponsorship for the years 2007-2008.

References

[1] www.analog.com/processors/blackfin/
[2] www.atmel.com/products/Diopsis
[3] M. D. Razdan, R. Brace, K. Smith, "PRISC software acceleration techniques", Proc. IEEE 1994 Intl. Conf. on Computer Design: VLSI in Computer & Processors, pp. 145-149.
[4] M. D. Razdan, R. Smith, "A high-performance microarchitecture with hardware programmable functional units", Proc. of MICRO-27, Nov. 1994, pp. 172-180.
[5] Scott Hauck, Thomas W. Fry, Matthew M. Hosler, Jeffrey P. Kao, "The Chimaera Reconfigurable Functional Unit", IEEE Trans. on VLSI Systems Vol. 12 , Issue 2 , Feb. 2004, pp. 206-217.
[6] H. Schmit, D. Whelihan, A. Tsai, M. Moe, B. Levine, R. Reed Taylor, "PipeRench: A virtualized programmable datapath in 0.18 micron technology", Proc. of the IEEE Custom Integrated Circuits Conf., 2002, pp. 63-66.
[7] H. Schmit "Incremental Reconfiguration for Pipelined Applications", IEEE Symposium on FPGAs for Custom Computing Machines.
[8] M. Vesterbacka, "A 14-transistor CMOS full adder with full voltage-swing nodes", SiPS 99, IEEE Workshop on Signal Processing Systems, pp.713-722.
[9] Fayed, A.A. Bayoumi, M.A. "A low power 10-transistor full adder cell for embedded architectures" Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE Intl. Symposium
[10] L. Di Nunzio, S. Fasciani "Implementation of DRM on DIOPSIS processor", Int. Rep. n. 32 Dept. of Electronic Eng., Univ. of Rome Tor Vergata (in italian)