A RECONFIGURABLE 2D CONVOLVER FOR REAL-TIME SAR IMAGING

Antonio G.M. Strollo, Ettore Napoli, Davide De Caro, Giacinto P. Saggese

Dept. of Electronics and Telecommunications Engineering, Univ. of Naples – "Federico II"
Via Claudio 21, 80125 Napoli, Italy - Fax: +39 081 5934448 - Email: etnapoli@unina.it

Abstract

A novel architecture for real-time Synthetic Aperture Radar signal processing that achieves real-time processing by using a recently proposed Signum Coded algorithm and time domain processing, is presented. New architecture is completely parallel and can be dynamically reconfigured in order to use different dimensions of data and filter matrices. A standard-cell VLSI implementation is presented and its performances are evaluated through circuit simulations.

1. INTRODUCTION

Synthetic Aperture Radar (SAR) is able to collect high-resolution heart images from both airborne and space born platforms. Main drawback is the huge post-processing of SAR signal needed to get the final image. Although some architectures for SAR data processing have been developed to speed-up the processing time [1,2], most of the approaches proposed to date are unable to generate the final image in real-time.

A new algorithm for time-domain SAR data processing ideally suited for direct implementation in a VLSI system has been proposed in [3-6]. In this approach, named one-bit coded SAR or Signum-Coded SAR (SC-SAR), both raw signal and reference function are coded with a single bit. The 2D correlation of the received echo data (the raw signal) with the point scatterer response of the SAR system (the reference function), is performed on the resulting binary sequences. Due to the reduction of the number of bits of the SAR data, SC-SAR allows significant reduction of CPU time with no loss of quality on the final image.

In [7-10] an innovative architecture for the elaboration of the SC-SAR algorithm based on the serial accumulation of 1D convolutions has been proposed. The circuit heavily reduces the complexity of previously proposed approach [11] using systolic arrays in order to achieve the real-time operation.

The need of multiple operating modes of the SAR sensor, and the changing of flight parameters, require a SC-SAR processor able to vary not only the content of the filter matrix, but also the data and filter matrices dimensions. Furthermore the reconfigurability of a 2D convolver is often required in many applications [12].

In this paper an architecture, which can be dynamically reconfigured in order to use different matrices dimensions, is presented. The novel structure is based on a completely parallel implementation of the SC-SAR algorithm that exploits pipelining for the correct synchronization of data in the 2D convolver.

2. SC-SAR ALGORITHM

SC-SAR processing can be reduced with no loss of generality to the problem of four 2D real convolution between SAR data (the raw signal) and the reference function (the filter) [6,7].

In the following we will assume filter dimension $Q \times P$ (Q rows and P columns), and image dimension of N bits for each column. Data column frequency is the pulse repetition frequency *PRF*.

As both reference function and data values are SC-coded, each sample can assume only two different values (-1, +1), and the same happens for their product.

If we use the following coding: -1 $\rightarrow$ logical 0 and +1 $\rightarrow$ logical 1, each product reduces to a simple XNOR operation between reference function and data bits.

Let us now consider the problem of summing the $R = Q \times P$ products. If X products are equal to +1 and R-X are equal to -1, then the total sum is:

$$S = 2X - R \qquad (1)$$

As a consequence, the sum is achieved by counting the number of products having a logical 1 value, multiplying the result by 2 (that is, including a 0 LSB) and subtracting a fixed offset which depends on the filter dimensions P and Q.

Thus, the main problem in performing the binary convolution is in calculating the elementary products and in counting the elementary products equal to 1.

3. ARCHITECTURE

The circuit for the parallel computation of 1D convolution between data vector and filter vector, named Σ_1 in the following, is composed of a set of XNOR gates that evaluate the elementary products of data and filter bits, and a binary compressor that counts the number of product bits equal to 1.

A simplified block diagram of the processor for the computation of 2D convolution is shown in Fig.1. The system computes:

$$Y_{n,m} = \sum_{i=0}^{Q-1} \sum_{j=0}^{P-1} H_{i,j} \cdot D_{i+n,\,j+m} \qquad (2)$$

where H is the filter matrix, D is the data matrix and Y is the convolution result matrix. In Fig.1, we have assumed that the data arrive from the SAR sensor serially by columns through the input *Serial_in*. The *SC-SAR_out* produces, serially and ordered by columns, the elements of the result matrix Y, after a latency of Q clock ticks.

The basic building blocks in Fig.1 are: the Data Formatter, the Σ_2 block and the Corrector.

The Data Formatter ensembles the serial input data and sends them in a suitable format to the Σ_2 block that is the core computational module of the whole SC-SAR processor. To reach this goal, the Data-Formatter receives the serial data through the *Serial_in*, and produces, at the k-th clock tick, the vector $Data\_out(k) \equiv [x_0(k), \ldots, x_{P-1}(k)]$:

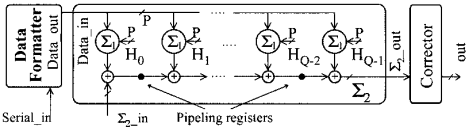

Fig.1 – Proposed SC-SAR processor.Data-Formatter reorders data bits for the computation through Σ_2 block.

0-7803-7057-0/01/$10.00 ©2001 IEEE.

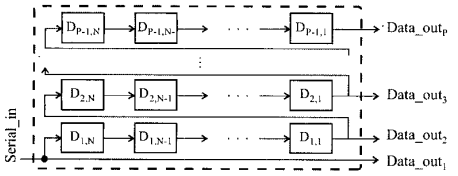

Fig.2 –Data Formatter implemented with flip-flop.

$$x_j(k) = D_{n,m} \qquad n = k \bmod N,\ m = j + \left\lfloor k/N \right\rfloor \qquad (3)$$

According to this equation, in subsequent clock ticks, partial rows of D matrix are reproduced on *Data_out* signal.

The Σ_2 block has two inputs: *Data_in*, that is connected to Data Formatter output, and Σ_{2_in} that is used for the reconfigurability of the convolver (see Section IV.1). For the time being we will consider $\Sigma_{2\_in} = 0$. The Σ_2 block is composed by Q 1D correlators. The analysis of the pipelined structure depicted in Fig.1, shows that the circuit computes the *SC-SAR_out* as the sum of 1D convolutions between the rows of H matrix and the last Q rows of the D matrix that have been sent in the previous Q clock ticks.

The Corrector block is used to subtract a fixed offset, as required by equation (1).

The proposed architecture (Fig.3) realizes a 2D convolution exploiting a completely pipelined, two dimensional, datapath that works at a single frequency equal to the SAR bit rate. Previous implementations of SC-SAR processor are based, instead, on a serial accumulation of results produced by a 1D convolver [7-10]. This requires different clock signals with frequencies multiple of SAR sensor bit rate ($N \times PRF$) with additional hardware and power dissipation.

A possible implementation of the Data Formatter of Fig.3, is depicted in Fig.2: it uses $(P\text{-}1) \times N$ flip-flops connected in a single shift register and one output is taken every N flip-flops.

It is possible to demonstrate that the circuit of Fig.2, implements the Data Formatter of Fig.3, according to equation (3), with an initial latency of $(P\text{-}1)\cdot N$ clock ticks.

Unfortunately this implementation requires a large number of flip-flops, since N is of the order of 1000 and P is of the order of 100. It is worth pointing out that many other convolver circuits, used in different applications, present the same problem requiring large memories for data reordering according to convolution definition. For example more than 50% area of the circuit proposed in [12] is used by the flip-flops for the data reordering operation.

In this paper a more efficient implementation of Data Formatter, based on a RAM, is presented. The technique makes it possible reordering of data required by the Σ_2 block, with a remarkable reduction of area and power dissipation with respect to the implementation of Fig.2.

To explain Data-Formatter operation, let us indicate with *Mem[i]* the i-th word of a RAM memory composed of N words P-1 bits long. The algorithm implemented by RAM Data Formatter is the following: on every rising edge of the clock, the i-th word of the memory (*Mem[i]*) has to be updated, by shifting its content and inserting the *Serial_in* bit. In the same clock tick, the $(i+1)$-th memory location has to be emitted on *Data_out*. The implementation of this algorithm requires three memory operation per clock cycle, one read and write on *Mem[i]* and one read on *Mem[i+1]*.

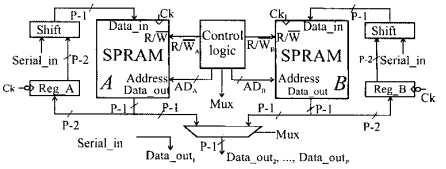

Fig.3 – RAM based Data Formatter. The shift blocks are simple hard-wired 1 position shifters.

The number of operations per clock cycle can be easily reduced to two, considering that the operation "read *Mem[i]*" is the same of the operation "read *Mem[i+1]*" performed on the previous clock cycle.

Assuming to use a memory executing only one operation per clock cycle (single ported RAM - SPRAM), that is synchronized on the rising edge of the clock, the proposed schematic of the RAM based Data Formatter is shown in Fig.3.

The Data Formatter has inputs and outputs synchronized with the rising edge of the clock. Internally it has two memory blocks, *A* and *B*, activated by the rising edge of the clock (synchronous memory). The SPRAM *A* contains even rows and *B* contains odd rows. In this way the two memory operations per clock cycle required by the algorithm, can be realized using one memory operation per clock cycle on *A* block and one memory operation per clock cycle on *B* block, as shown in the operation cycle reported in Fig.4.

The control logic in Fig.3 provides the appropriate timing to drive the memory address buses and Read/Write signals. The control logic algorithm simply realizes the operation cycle of Fig.4. Its implementation requires a simple counter modulo N with few other logic gates and flip-flops. The registers Reg_A and Reg_B, depicted in Fig.3, break the feedbacks around memories that is needed to realize the read/shift/write operation; in this way it is possible to guarantee the respect of memory timing constraints.

4. RECONFIGURABLE PROCESSOR

4.1 Reconfigurable 2D-convolver

A previous work [12] obtains the convolver reconfigurability using an oversized datapath operating on a larger matrix filter and inserting zeros to delete undesired rows and columns. This technique can not be applied to the SC-SAR processor because the signum-coding permits only ±1 values. We present a different reconfigurability approach that overcomes this limitation. New approach is based on the following considerations.

The schematic shown in Fig.5 demonstrates how two 2D elementary convolvers, which operate on $Q \times P$ filter matrices, can be interconnected in order to realize a $Q \times 2P$ convolver.

Each elementary convolver is implemented as the Σ_2 block of Fig.3. It receives as input two halves of each data matrix row, producing the convolution relative to its portion of the total filter matrix. The adder sums the two partial convolutions to obtain the total result.

	CK k	CK k+1	CK k+2	CK k+3
Operation on A	SHIFT/WRITE Mem[i]	READ/EMIT Mem[i+2]	SHIFT/WRITE Mem[i+2]	READ/EMIT Mem[i+4]
Operation on B	READ/EMIT Mem[i+1]	SHIFT/WRITE Mem[i+1]	READ/EMIT Mem[i+3]	SHIFT/WRITE Mem[i+3]

Fig.4 - Operations cycle for the circuit of Fig.3.

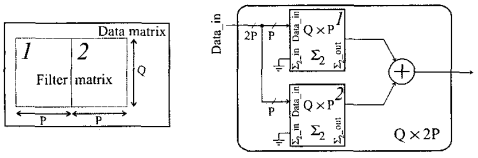

Fig.5 – A Q×2P convolver using Q×P elementary convolvers.

In Fig.6 it is shown that a $2Q \times P$ convolver can be realized using two elementary $Q \times P$ convolvers.

The first elementary convolver elaborates the convolution relative to its portion of the total filter matrix, presenting this result on the Σ_{2_in} input of the second block. This block elaborates its convolution and adds the result of the previous block that is provided through its Σ_{2_in} input. Note that the two partial convolutions, computed by the 2 elementary convolvers of Fig.6, are available at different clock ticks. The pipelining registers between Σ_{2_in} and Σ_{2_out} of block 2 (see Fig.3) guarantee that coherent partial convolutions are added together for producing the total output.

The combination of above mentioned properties makes it possible the realization of reconfigurable 2D convolvers. In Fig.7 a reconfigurable architecture that uses 16, $Q \times P$, elementary convolvers is shown. The convolvers can be dynamically interconnected in different way, following the above mentioned approaches, in order to obtain the configurations reported in Tab.1.

In Fig.7 the Controller is a combinational block that acts on the multiplexers control signals (s_i and t_i), in order to create the appropriate datapath among the elementary Σ_2 convolvers, according to the configuration coded in the *Rows* and *Columns* inputs. The Data Dispatcher is also a combinational block that properly subdivides each row *Data_out*, produced by the Data Formatter, and routes each partial row to the correct elementary convolver. Note that the final addition stage also takes into account the constant that has to be added to correct the convolution result.

For example the $6Q \times 2P$ configuration can be obtained switching the multiplexers so that the blocks 3-8 and 11-16 are connected respectively in two series ($s_3 = s_4 = s_5 = s_6 = 1$), while s_2 is set to 0 in order to separate the series from the unused elementary convolvers. The

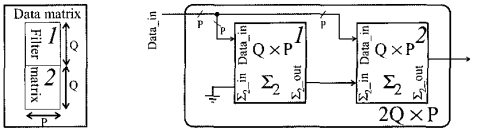

Fig.6 – A 2Q×P convolver using Q×P elementary convolvers.

multiplexers controlled by t_1 e t_3 are switched to zero ($t_1 = 00$ and $t_3 = 00$) while the others propagate the outputs of the two series ($t_2 = 0$ and $t_4 = 0$) to the adders so that the whole $6Q \times 2P$ convolution result can be calculated.

Another possible configuration example is the $2Q \times 8P$. In this case the multiplexers are configured in order to connect all Σ_2 blocks in 8 series each composed of 2 blocks (the series 1-2, the series 3-4, etc.). The adders Add_1 , … , Add_4 sum the 8 partial convolutions, produced by the 8 series, in order to obtain 4 addends that are propagated through the multiplexers to the final addition stage to achieve the convolution result.

4.2 Reconfigurable data-formatter

To realize a completely reconfigurable SC-SAR processor, also the Data Formatter of Fig.3 has to be made reconfigurable, allowing to dynamically change the parameters P and N. This is possible using the proposed schematic of Fig.4, where the two RAM memories (each RAM is composed of $N/2$ words of P-1 bits) are reconfigurable and the Control logic uses a programmable counter with variable module equal to N.

A simple way for realizing a reconfigurable RAM module is shown in Fig.8.

The proposed solution allows 3 different configurations ($2^H \times 4W$, $2^{H+1} \times 2W$, $2^{H+2} \times W$) with no memory waste. The *Mode* signal is assumed to encode the desired configuration. The Address Generator block, according to the *Mode* input and to two bits of *Address*, computes the control signals for tristate buffers and enables the elementary $2^H \times W$ memory blocks.

5. CIRCUIT PERFORMANCES

Using the proposed architectures of the 2D convolver and the Data Formatter, we have designed up to the layout level the SC-SAR processor circuit with CSI AMS 0.35µm standard-

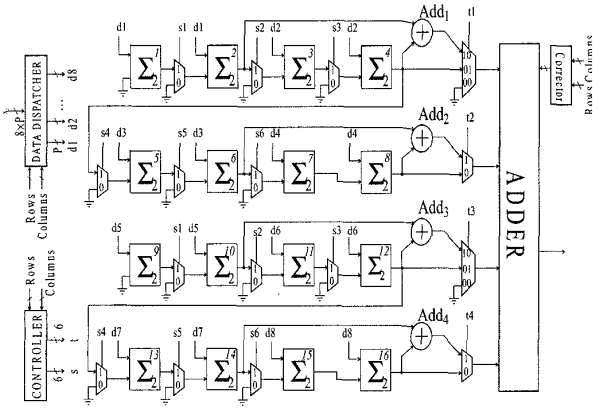

Fig.7 – Proposed architecture of the reconfigurable 2D convolver.

	2P	3P	4P	5P	6P	7P	8P
2Q	✓	✓	✓	✓	✓	✓	✓
3Q	✓	✓	✓				
4Q	✓	✓	✓				
5Q	✓						
6Q	✓						
7Q	✓						
8Q	✓						

Tab.1 – Possible configurations using the circuit of Fig.5.

	SC-SAR Processor		DataFormatter
	2D Conv.	Data-Formatter	(Fig.2)
Area (mm^2)	52.5	32.8	142.6
Power (W/MHz)	0.084	0.038	0.91

Tab.2 – Performances of proposed circuits

cell technology and 3.3V power supply. The circuit includes 16 elementary 2D convolver with filter dimensions $P = 32$ and $Q = 32$, connected as shown in the schematic of Fig.8, and a Data Formatter that uses a total of 8 elementary block memories 1024×64. The whole system can be reconfigured in 17 operating modes, as reported in Tab.1, with 256×256 maximum filter dimensions. The Data Formatter is able to reorder the input data received from the SAR sensor, with $P = 256$, $N \leq 2048$ or $P = 128$, $N \leq 4096$ or $P = 64$, $N \leq 8192$. The circuit layout has been obtained using the tool SILICON ENSAMBLE of CADENCE. A portion of the layout is shown in Fig.9.

The timing performances of the circuit, including parasitics, have been obtained through PEARL simulations, while power consumption have been evaluated by SPECTRE. The whole circuit is able to operate up to 40MHz, whereas the Data Formatter can work at a clock frequency greater then 100MHz. In Tab.2 area occupation and power dissipation are reported. For a comparison, area and power dissipation for the flip-flop based implementation of the Data Formatter (see Fig.2) are also reported.

The analysis of the data shown in Tab.2 clearly demonstrates the effectiveness of RAM Based Data-Formatter. Note that greatest part of total power and area is due to the 2D Convolver (68% of total power and 61% of total area). It is worth pointing out that only the 1.2% of the area occupation of the circuit is devoted to the reconfigurability logic, showing the effectiveness of the proposed approach.

6. CONCLUSIONS

A novel architecture for real-time Synthetic Aperture Radar signal processing has been presented.

With respect to previously proposed architectures, the new structure is completely parallel and can be dynamically reconfigured in order to use 17 different dimensions of data and filter matrices. A standard-cell VLSI implementation has been presented and its performances have been evaluated through circuit simulations showing the effectiveness of the proposed approach (only 1.2% area overhead due to the reconfigurability).

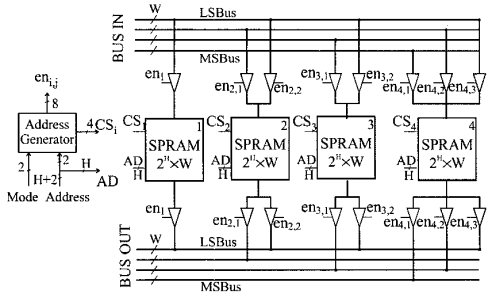

Fig.8 – Reconfigurable memory block for 3 different configurations with no memory waste.

7. ACKNOWLEDGES

Authors would thank Danilo Palomba for his help in VHDL description and simulations of the SC-SAR processor.

8. REFERENCES

[1] G.Franceschetti et al., "An efficient SAR...", IEEE Tran. Aero. Elect. Sys., vol.27, No.2, pp.343-53, 1991.

[2] J. Dall et al., "Real-time processor for ...", IEE Proc. F, vol.139, No.2, pp.115-21, 1992.

[3] G. Franceschetti et al., "Processing of signum-coded ...", IEE Proc. F, vol.138, No.3, pp.192-198, 1991.

[4] G. Alberti et al., "Time-domain convolution ...", IEE Proc. F, vol.138, No.5, pp.438-44, 1991.

[5] G. Franceschetti et al., "Time-domain processing ...", Proc. of the IGARSS Conf., pp. 283-286, Helsinki, 1991.

[6] G. Franceschetti et al., "An architecture ...", Proc. of the ISPRS Conf., pp. 527-532, Washington, D. C., 1992.

[7] A.G.M. Strollo, E. Napoli, C. Cimino, "A VLSI processor ...", Proc. of the ICECS Conf., pp.1631-1634, 1999.

[8] A.G.M. Strollo et al, "A VLSI architecture for ...", Proc. of the ISSSE Conf., pp.282 –286, 1998.

[9] A. Strollo et al., "A novel VLSI architecture ...", Proc. of the IGARSS Conf., pp. 515 – 517, 1999.

[10] A. Strollo et al, "A novel time-domain processor for real time SAR operation", PIERS'98 Conference 13-17 Luglio 1998 Nantes, pp.627.

[11] G. Cappuccino et al., "Design and ...", IEE Proc. Radar Sonar and Navig., vol. 143, No. 4, pp. 261-267, 1996.

[12] B. Bosi, G. Bois, Y. Savaria, "Reconfigurable Pipelined...", IEEE Trans. On VLSI Systems, vol.7, no.3, Sep. 1999, pp.299-308.

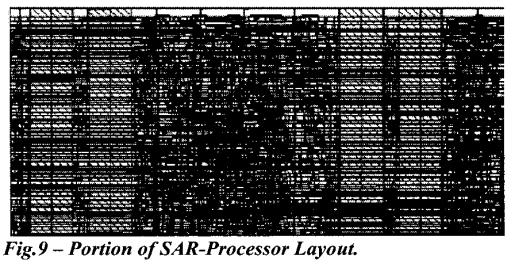

Fig.9 – Portion of SAR-Processor Layout.