ERS-1 SAR processing with CESAR.

by

Einar-Arne Herland
Division for Electronics
Norwegian Defence Research Establishment
P.O.Box 25, N-2007 Kjeller, Norway

Abstract

A vector processor called CESAR (Computer for Experimental Synthetic Aperture Radar), is now in the final stage of development at the Norwegian Defence Research Establishment. It is connected to a supermini computer, and has a maximum performance of 320 Megaflops. This is achieved through parallelism and pipelining, and the processor is fully programmable. This paper describes the architecture of CESAR and the implementation of a set of algorithms for processing of images from the ERS-1 synthetic aperture radar.

1. Introduction.

Processing of SAR raw data into images is computationally very demanding, and regular use of SAR images requires a very high processing capacity. With a series of satellite-based SARs planned for the next decade, efficient utilization of the systems requires that suitable processors be available soon. The main objective for developing CESAR is to make possible the use of the coming SARs for surveillance of the Norwegian economic zone, and the first of the available platforms will be ERS-1 to be launched by the European Space Agency (ESA) in the first half of 1990. A prototype of CESAR will be ready by the end of this year, but through extensive simulation and testing of the parts that are already completed, a very accurate estimate of the performance can be given.

2. The CESAR system.

2.1 System overview.

An earlier version of CESAR has been described in [1]. Figure 2-1 shows a block diagram of CESAR. The processor is connected to a host computer via the data port module (DAP), which handles the transfer of both data and programs to CESAR. All programming is done on the host computer, which also takes care of communication with surrounding systems, like high density digital recorders (HDDR), discs, graphic displays and output media. This is done via the multiport memory (MPM) on the host computer.

Application programs run in the control processor (CP), which is a Motorola 68020 microprocessor. The application program is written in CESAR

PASCAL, which is an extension of standard PASCAL. This is necessary in order to allow parallel processes to run in CESAR, like data transfers between memory modules, and the actual processing. A special extension is also used to allow the CP to refer to data residing outside its own local memory. The CP also controls the other parts of CESAR by sending control information over the VMEbus, which in addition can be used for low capacity data transfers.

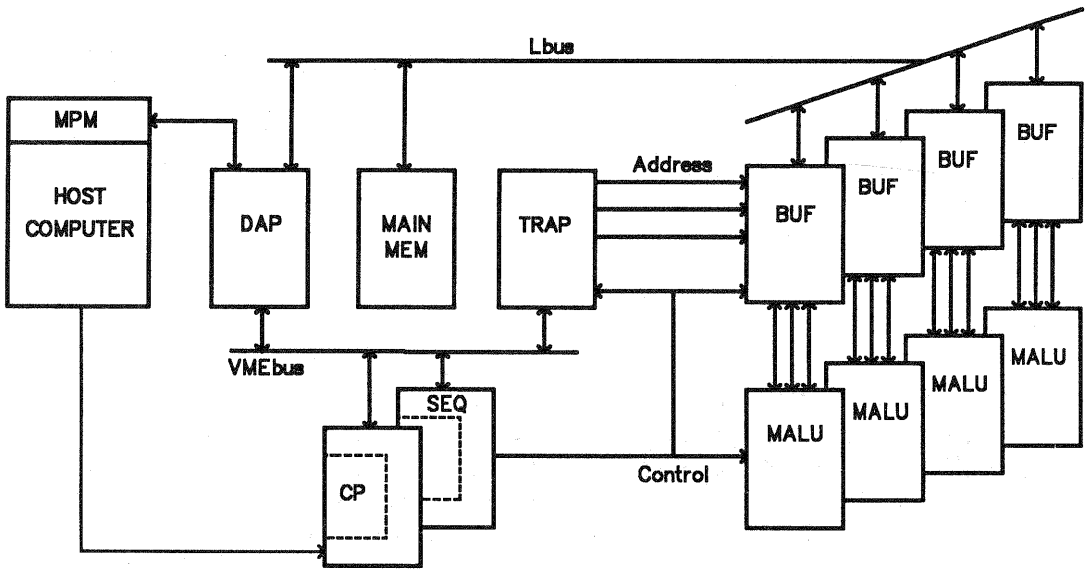

Figure 2–1 CESAR block diagram.

The actual processing is done in the microprogrammable arithmetic–logic unit (MALU). This module is described in more detail in section 2.2. The CP can also be used for data processing on small amounts of data if it has spare capacity, or if a particular task is not suitable for implementation on the MALU. The MALU gets its input data from and stores its output data in a buffer memory (BUF). BUF is divided into three banks of equal size, and when running a program in the MALU, two of the banks are connected to the MALU input, and the third receives the output data. The connections between the BUF data banks and the MALU I/O–lines is under full program control. CESAR contains four identical, parallel sets of MALU/BUF, which work completely synchronously, always with identical programs, but on different data sets. Addressing in the BUFs is performed by the triple address processor (TRAP), where "triple" refers to the three data banks in BUF. The same addresses are fed to all four BUFs. Synchronization of MALU, BUF and TRAP is performed by the sequencer (SEQ).

The main memory module (MAIN MEM) is used to store intermediate data, and in the case of SAR processing it is used as corner turn memory. Input data from MPM is typically fed directly to the BUFs and output data is transfered directly from the BUFs to MPM, while the MAIN MEM is used for data transfers to and from the BUFs. MAIN MEM and the BUFs are connected via the Lbus, which is controlled by the DAP. The DAP can transfer data between any two memory modules connected to it.

Data transfer capacities are 40 Mbytes/sec locally on the Lbus, 10 Mbytes/sec for transfers from MPM to Lbus, and 6 Mbytes/sec from Lbus to MPM. Each of the three I/O-lines between BUF and MALU transfers data at a rate of 40 Mbytes/sec. In the present configuration MAIN MEM contains 32 Mbytes of dynamic RAM, and each BUF has 6 Mbytes of static RAM equally divided on three banks. CESAR thus contains 56 Mbytes of memory in addition to CP's local memory, which is mainly used for storing system information.

In order to run an application on CESAR, four sets of programs must exist. Three of them are inside CESAR, and the fourth is running in the host, taking care of data communication with peripheral devices as mentioned above. It is also possible to do actual data processing on the host. The host programs are typically written in FORTRAN. The three program sets in CESAR are the CP program and the TRAP and MALU programs. A library of MALU and TRAP programs exists, and if this library is sufficient, only the CESAR PASCAL program has to be written, and the MALU and TRAP programs are implemented as function calls. Both compilers and simulators exist for MALU and TRAP in case a user wants to write programs for these modules. There also exists a simulator for the CP program, whereby bottlenecks in the program can be detected and performance estimated.

2.2 The microprogrammable arithmetic-logic unit, MALU.

The actual computations are performed by the MALU. This module consists of a matrix with 128 identical S-elements, connected as a sylinder, as shown in figure 2-2. The right-hand view shows the data matrix and the interconnections between the 16 by 8 S-elements, while the left-hand view shows how the matrix is connected into a cylinder. Data is fed to the top of the cylinder on parallel form, where it is converted to serial form and clocked bitwise down into the cylinder. On the output the reverse operation is performed. Two 32-bit data words enter and one 32-bit word exits the cylinder every 100 ns when a program is running.

Each S-element is a single VLSI chip, and it has a repertoire of 32 instructions implementing floating point, fixed point and logical operations. It works with bit-serial data, and uses on-chip pipelining to implement the instructions. It has 4 input lines, two from above and one from each of the horizontally neighboring elements, and 6 output lines, two vertically downwards, two horizontally and two diagonally downwards. For the elements in the bottom

row only one output exists vertically. The routing of data along these I/O-lines is programmable, and data can move horizontally and downwards through the matrix. On the top of the cylinder data is input from one or two of the data banks in BUF, and the output data from the bottom row is written back to the third data bank of the BUF.

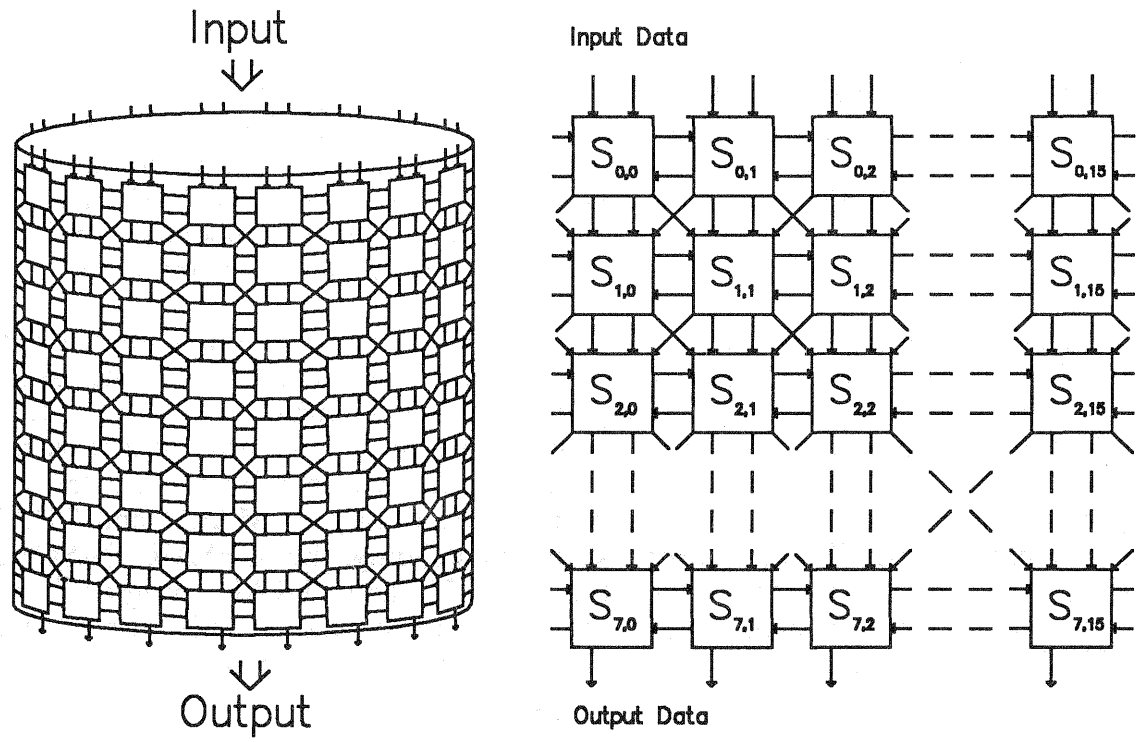

Figure 2-2 The systolic array MALU.

When MALU is running a program, each S-element performs a single operation with a fixed routing combination given by the instruction word. The S-element contains delay registers in order to insure synchronous arrival of the operands, and it can store 32 different instruction words, which gives rapid change of MALU program. By combining the operations in different S-elements, complex algorithms can be implemented by a single pass through the MALU. Division is not implemented in the present version of MALU.

The actual throughput depends on the algorithm implemented. If a simple addition of two vectors is performed, only one cylinder row is used, and the throughput is accordingly only 1/8 of maximum performance. The more complicated the operation, the better the performance. Figure 2-3 shows how a radix-2 FFT butterfly is implemented. It uses three rows, and the rest of the rows are not shown, since data only is routed directly through them. Since the width of the algorithm is four S-elements, four butterflies fit around the cylinder. The W's are twiddle factors, while X and Y are the complex input

data.

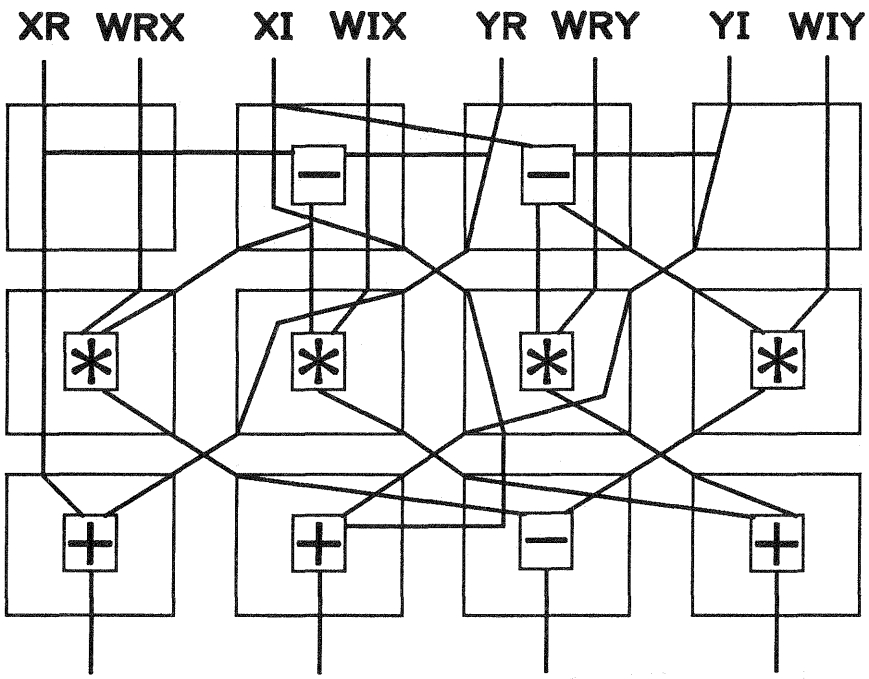

Figure 2–3 Radix–2 butterfly.

3. SAR processing on CESAR.

The implementation considered here is the one which is going to be used for processing of SAR images from ESA's ERS–1 satellite. This is a C–band SAR, which is reflected in some of the algorithm parameters. It has less range curvature than L–band SARs, but more than an X–band SAR. Yaw steering will also reduce the range cell migration (RCM). In spite of these differences, the structure of the algorithms would be much the same for all wavelengths. Highly squinted systems, however, may require different processing algorithms.

3.1 Processing algorithm.

The algorithm used here is the socalled range/doppler algorithm, where the two–dimensional compression of the target signal histories is divided into two one–dimensional operations, range compression and azimuth compression. Range dependence of the azimuth signal is handled by changing azimuth filters across the swath, while range cell migration correction (RCMC) is done in the azimuth frequency domain. This is valid for SARs with small squint angles.

Processing is done in terms of reference scenes of 100 km by 100 km. In order to achieve this, the scene is broken down into a number of smaller

processing blocks. This is described in more detail in section 3.2. The algorithm described in this section applies to a single processing block.

Figure 3–1 shows a block diagram of the algorithm. The first step is to unpack the raw data from the packed 2 x 5 bit per complex sample used in the input, and convert to 32–bit floating point format used in the computations. The next step is to range compress the data, which consists in fourier–transforming the range vector, multiplying with a reference filter, and inverse transforming the result. At this stage the data is converted to 16–bit integer format and intermediately stored in MAIN MEM. By properly scaling the range reference filter, the full dynamic range of the range compressed data is retained. The conversion is done to save memory space and data transfer capacity on the Lbus.

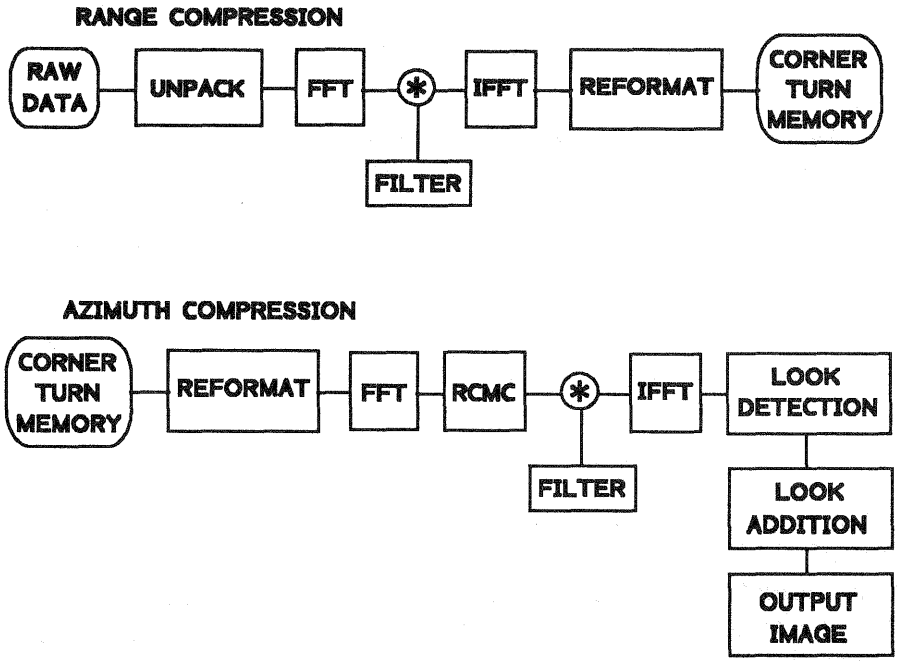

Figure 3–1 Block diagram of SAR algorithm.

When the processing block has been range compressed, azimuth compression is performed. Corner turning of the data is done by reading back the range compressed data from MAIN MEM in azimuth direction. The first operation is to convert the data from 16–bit integer form back to 32–bit floating point format. The azimuth vectors are then fourier–transformed and RCMC is performed by interpolating in the transformed data in range direction according to the target trajectory. The straightened azimuth vectors are then multiplied with the look filters and inverse transformed. Look detection is done by taking the absolute value of the compressed data, and the looks are added together. Overlapping parts from previous processing blocks are read back from MAIN

MEM and added to the present data. The part containing the full number of looks is then output to the host, and the incomplete part is written to MAIN MEM.

The processing chain described above is not complete, and it is only valid for a fast-delivery type product which is not geometrically corrected. In addition to what is shown above, doppler centroid frequency estimation is performed, but only on a small part of the data, and if orbit data is not available, autofocusing must be applied to estimate the azimuth doppler rate. The processing necessary to implement these steps is only a few percent of the total processing load. Geometric correction of the compressed image, however, is a major operation which can be quite time consuming depending on the level of accuracy desired.

3.2 Implementation.

The input data matrix for a 100 km by 100 km scene consists of about 6000 by 25000 complex samples, in range and azimuth, respectively. This is divided into 4 by 35 processing blocks of identical size. Figure 3-2 shows this division into processing blocks.

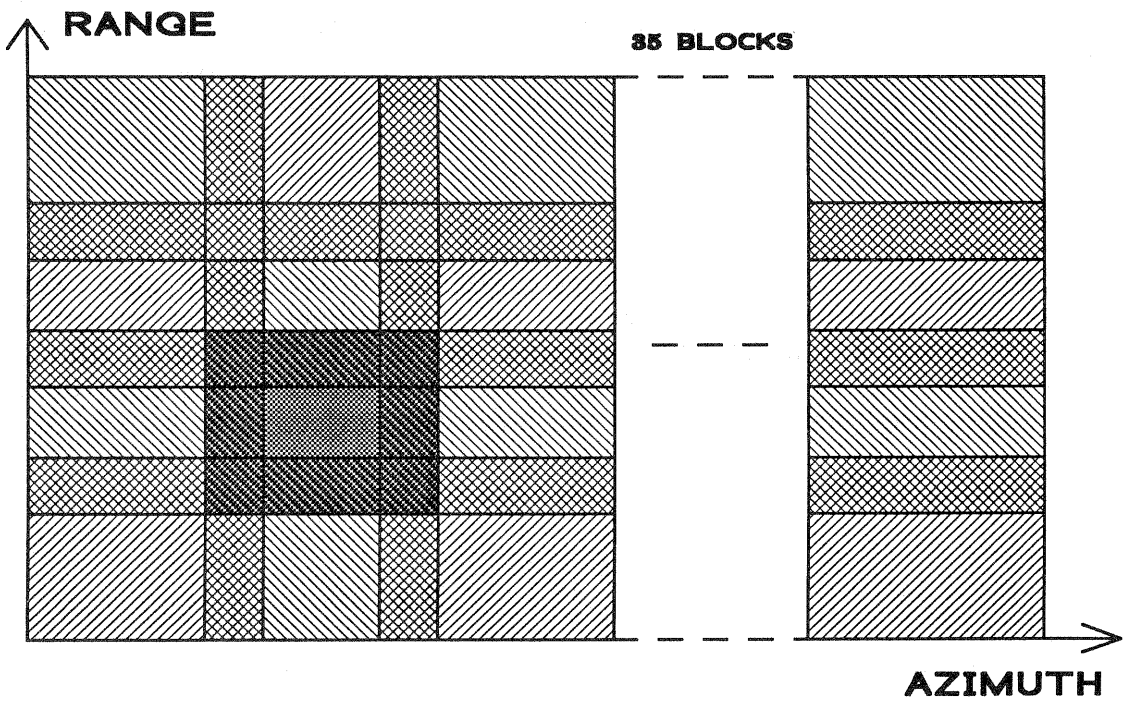

Figure 3-2 Division of scene into processing blocks.

CESAR processes the four blocks lying at the same azimuth position in parallel, with one block for each of the four MALU/BUF sets. The block length is 2048 in range and 1024 in azimuth, suitable for multilook processing. Range overlap is approximately 700 and azimuth overlap is 256, corresponding to one out of four looks. Since a single block is too big to fit into one BUF, it is processed by dividing it into smaller blocks of f.ex. 32 vectors at a time. When possible, these vectors are considered as one long vector, thereby effectively eliminating the importance of the pipeline tails in MALU.

A set of different MALU programs is needed to implement this algorithm. If a specific operation, e.g. unpacking of raw data, cannot be performed in one pass through MALU, more passes will be needed, and the programs for these passes may or may not be different from the programs in previous passes. In the case of FFT, the number of passes depends on the vector length. One pass implements a single butterfly, which means that a radix-4 1K FFT needs 5 passes, but in this case all passes use the same program. A 2K FFT needs the same 5 passes and in addition a radix-2 pass, which uses a different program. Multiplication of two complex vectors is done in a single pass, while unpacking of raw data needs two passes. The reason for this is the cyclic nature of the input. Since the cylinder input row has 16 S-elements, the input data must also be cyclic with a period of 16. If unpacking is to be done in a single pass, the input will lose its cyclic nature. Vector addition is done in one pass, while the square root in the detection operation needs 3 passes with different programs. It turns out that the total number of MALU programs needed for SAR fast delivery images is 10-15, which means that all the programs can be downloaded into the MALU before processing starts.

3.3 Performance.

Since MALU works synchronously, it is easy to calculate the performance of a specific algorithm. The time used is proportional to the vector length, except the pipeline tails at the beginning and end of the vector, which only depend on the algorithm, and is longer the more complex the algorithm. The vector lengths used here are long enough to justify neglecting the tails.

Unpacking a 2K vector of raw data requires two passes, where the first pass is shorter than the second. The total time is

$$25000(512 \times 200 \text{ ns} + 2048 \times 200 \text{ ns}) = 12.8 \text{ s}$$

A 2K FFT needs 6 passes through the MALU, which gives

$$2048 \times 6 \times 200 \text{ ns} = 2.5 \text{ ms}$$

for a single MALU. CESAR thus uses 0.6 sec on a 2K FFT. A 1K FFT is performed in 0.25 sec on the average in CESAR. Filter multiplication takes 0.2 ms for a 1K vector, and conversion from 32-bit format to 16-bit format the

same. Unpacking, range compression and format conversion accordingly require 160 seconds for the whole scene.

Azimuth compression is performed on 35 x 5400 1K vectors = 189000 1K vectors. The forward and inverse FFTs and filter multiplications are done in 110 seconds, if it is assumed that the multilook processing requires the same time as onelook processing. If range cell migration correction uses a four-point interpolation filter in range, this operation consumes 40 seconds for the whole scene. When reformatting of the input data, look detection and addition are added, the total time for azimuth compression will be approximately 175 seconds.

The calculations above assume that data is always available in the BUFs. This is achieved by using double buffering, where data I/O between BUF and the Lbus is done simultaneously with the MALU processing. The Lbus is able to feed the four MALU/BUF sets with data because many passes are applied to all the data before it is output to the Lbus again, and the data format is more compact for the Lbus data than for the MALU data. The goal for processing of a reference scene is 8 minutes, which will be achieved with the system described above.

4. Conclusion.

During the design of CESAR, SAR processing has been the main application area. The system can, however, be used in a wide range of application areas, notably signal and image processing. It is a vector processor, and is especially efficient for complicated computations where many operations are to be performed on the data.

References

[1]:
V.S.Andersen, T.Haugland, O.Søråsen: *CESAR – A programmable systolic array multiprocessor system.* The first International Conference on Supercomputing Systems, Dec 1985, St. Petersburg, FL.