Processor Architectures for Synthetic Aperture Radar

by

Peter G. Meisl

B.A.Sc. Electrical Engineering, University of British Columbia, 1990

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

DEPARTMENT OF ELECTRICAL ENGINEERING

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

March 1996

Department of Electrical Engineering

The University of British Columbia
Vancouver, Canada

Date April. 24/96

Abstract

This thesis examines processor architectures for Synthetic Aperture Radar (SAR). SAR is a remote sensing technique that requires large amounts of computation and memory to form images. Processor architectures are sought that exhibit high performance, are scalable, are flexible, and are cost effective to develop and build. Performance is taken to be the primary figure of merit.

The three facets of systems design, namely algorithm, technology, and architecture, are each examined in the process of finding the best architecture implementations. The examination of the algorithms is begun by reviewing SAR processing theory with the intent of summarizing the background for typical SAR processor performance requirements. A representative set of SAR algorithms is analyzed to determine and compare their computational requirements and to characterize them in terms of basic digital signal processing (DSP) operation types. The algorithm partitioning options for parallel processing are classified and compared.

The area of technology is addressed by surveying computing technologies that are relevant to SAR processing. The technologies considered cover computational devices, memory devices and interconnections.

The knowledge of SAR algorithms and computing technology are then combined to study design considerations for the memory and interconnection subsystems of SAR processors. The requirement for two dimensional access to large data arrays is found to be the main complicating factor in memory design. The judicious use of wide data path widths, caches, interleaving and fast memory is discussed as a solution to the memory latency problem. The applicability of the most commonly used interconnection networks is examined. Buses, meshes, and crossbars are all found to be effective in certain situations.

A classification of architectural approaches adapted to describe current and future SAR processors is used as the framework for the architecture selection. Feasible implementations of the architectural approaches are identified and their suitability for SAR is analyzed. Three approaches are identified as the most promising: networks of workstations, DSP chips, and field programmable gate arrays (FPGAs). A detailed examination is made of these three approaches. Four variations of the DSP approach are considered: general purpose DSPs, vector DSPs, a proposed optimised DSP, and digital filter devices. Each approach offers a different trade-off between performance and flexibility. The most cost effective architectures approaches are found to be those based on general purpose or vector DSPs. An original heterogeneous design is presented that combines the strengths of these two approaches.

Table of Contents

List of Tables

List of Figures

Symbols and Acronyms

γ	antenna elevation angle
λ	wavelength
θ	squint angle
θ_s	instantaneous squint angle
θ_v	antenna azimuth beamwidth
η	azimuth time variable
η_0	azimuth time at which sensor is closest to target
η_c	beam center offset time
$\Delta\eta$	exposure time
σ'	scatterer reflectivity
τ_p	pulse duration
τ_c	coherent integration time
B_D	Doppler bandwidth
B_p	processed Doppler bandwidth
c	speed of light
CVM	complex vector multiply operation
CVRVM	complex vector / real vector multiply operation
f_0	carrier frequency
f_η	azimuth frequency
$f_{\eta c}$	Doppler centroid
f_s	range sampling rate
Δf_τ	pulse bandwidth
FilterCC	filtering operation with complex data and complex coefficients
FilterCR	filtering operation with complex data and real coefficients
FilterRR	filtering operation with real data and real coefficients
FFT1d	one dimensional FFT operation
g_a	azimuth processing efficiency
g_r	range processing efficiency
GR	ground range
h	platform altitude
H	height of antenna
L_{az}	azimuth reference function length
L_r	range reference function length
K_a	azimuth frequency rate
K_r	range frequency rate
L	antenna length
n_u	azimuth matched filter update interval
N_{az}	size of data array in azimuth
N_{filter}	interpolation filter length
N_r	size of data array in range
N_{RC}	maximum range curvature

N_{RW}	maximum range walk
N_s	maximum azimuth shift between blocks
OP_{ref}	number of operations required to compute each point of azimuth matched filter
$p(\tau)$	baseband representation of transmitted pulse
r_0	range at time of closest approach
r_c	range when target passes through center of beam
δr	spatial range resolution
$R(\eta)$	distance between platform and target
R_m	range at mid-swath
Root	root operation
Round	rounding operation
RVS	real vector scalar operation
S_r	correlation replica
$s_{tx}(\tau)$	transmitted pulse
SR	slant range
t	range time variable
T_p	pulse repetition period
TB	azimuth time bandwidth product
v	platform velocity
$w(\eta)$	antenna pattern in azimuth
W_r	extent of radar beam in slant range
δx	spatial azimuth resolution

AC	azimuth compression
ALU	arithmetic logic unit
ARPA	Advanced Research Projects Agency
ASIC	application specific integrated circuit
ATM	asynchronous transfer mode
BPF	block floating point
CCRS	Canadian Center for Remote Sensing
CFFT	complex FFT
CISC	complex instruction set computer
CLB	configurable logic block
CN	common node
COTS	commercial off-the-shelf
CPU	central processing unit
CTM	corner turn memory
dB	deciBel
DEM	digital elevation model
DRAM	dynamic RAM
DSP	digital signal processing or digital signal processor
EDO	extended data out
ERS	earth resources satellite

FFT	fast Fourier transform
IFFT	inverse fast Fourier transform
FIR	finite impulse response
FLOP	floating point operation
FM	frequency modulation
FPGA	field programmable gate array
GP	general purpose
HDL	hardware description language
HW	hardwired
IC	integrated circuit
IIR	infinite impulse response
JPL	Jet Propulsion Lab
MAC	multiply accumulate
MB	mega-byte
Mb/s	mega-bits per second
MB/s	mega-bytes per second
MCW/s	mega-complex words per second
MC	multicomputer
MCM	multi-chip module
MDA	MacDonald Dettwiler and Associates
MIMD	multiple instruction multiple data
MISD	multiple instruction single data
MP	mulitprocessor
MPI	message passing interface
MW/s	mega-words per second
NASA	National Aeronautics and Space Administration
NOW	network of workstations
PCI	peripheral component interconnect
OP	operation (arithmetic)
PE	processing element
PL	pipeline processor
PRF	pulse repetition frequency
PVM	parallel virtual machine
RAM	random access memory
RASSP	rapid prototyping of application specific signal processors
RC	range compression
RCM	range cell migration
RCMC	range cell migration correction
RF	radio frequency
RFM	reference function multiply
RISC	reduced instruction set computer
RT	real-time
SAR	synthetic aperture radar

SIMD	single instruction multiple data
SIR	shuttle imaging radar
SISD	single instruction single data
SP	single processor
SPECAN	spectral analysis
SRA	sample rate adjustment
SRAM	static RAM
SRC	secondary range compression
VASP	versatile array signal processor
VLSI	very large scale integration
VME	versa module european
VSA	vector scalar architecture
WE	wave equation

Acknowledgment

I would like to thank my supervisors, Dr. M.R. Ito and Dr. I. Cumming, for their support. Many thanks are also due to all those in the High Performance Computing Lab who helped provide an enjoyable and constructive work environment. I also thank the Natural Sciences and Engineering Research Council of Canada, the University of British Columbia, and MacDonald Dettwiler and Associates for their financial support.

1 Introduction

1.1 Background

Synthetic Aperture Radar (SAR) is a remote sensing technique for obtaining high resolution images of the earth's surface. SAR sensors are typically flown in airplanes or satellites. Since SAR is a radar sensor, it has the advantage that it can collect images at night and through cloud cover. There has been great growth in the field of satellite SAR in the last few years. Until quite recently, only one SAR satellite had ever existed: Seasat. However, now several SAR-capable satellites are in operation with several more coming.

The study of SAR processing is particularly relevant to Canada's interests. Canada's vast geography has made it necessary to become a world leader in remote sensing. Canada has been at the forefront of research in SAR and SAR processing since the first SAR satellite. This was continued in 1995 with the launch of Canada's Radarsat satellite, which is a state-of-the-art SAR satellite.

One of the traditional problems with SAR as a remote sensing tool is the huge amount of signal processing that is required to form an image from the raw data. Originally, optical techniques were used to process SAR data. Optical methods were superseded by digital processing in the late 1970s. Digital SAR processing systems have usually been either very slow or very complex and expensive. However, technology has advanced a great deal since the first digital SAR processors were built. The processing power of computer components has been catching up with the requirements posed by SAR processing, opening up new possibilities for SAR processor designs. This thesis explores some of these new possibilities.

SAR processors are complex high performance signal processing systems. The designer of such a system must address each of the three fundamental facets of signal processing systems: algorithms, component technology, and architectures.

Algorithm

The characteristics of the SAR processing algorithm must be understood in detail. This includes the structure, computational requirements, and issues affecting the partitioning for parallel processing.

Technology

The current state of technology must be understood. This includes making a survey of the available components that might be used in building a SAR processor.

Architecture

The high computational requirements of SAR processing call for sophisticated computer architectures and parallel processing techniques. The various types of signal processor architectures must be examined and the most promising ones identified.

Each of these three areas offers the designer many choices and trade-offs. The system design process seeks an optimal combination of choices in each of the areas.

1.2 Objectives

The main objective of this thesis is to examine the alternatives for SAR processor architectures and find the best ones based on current technology. The goal is to come up with SAR processor designs that

- exhibit high performance,
- are scalable,
- are flexible, and
- are cost-effective to develop and build.

The approach used to meet these requirements is to look at each of the three areas of algorithm, technology, and architecture in turn, and then try to combine them into design solutions. Performance is used as the primary criteria in the examination of architectures.

Since there is a wide range of relevant computing technologies, a similarly wide range of architectures must be examined. No wide ranging up-to-date treatment of the alternatives for SAR processor architectures is currently available. This thesis fills that gap and provides a basis for continued work in SAR processor architectures.

1.3 Outline

The structure of this thesis is summarized below.

Section 1 provides the background, objectives and outline for the thesis.

Section 2 addresses the algorithm aspect of SAR systems design. An overview of SAR processing is given with an emphasis on the issues that affect the performance requirements of a high-speed processor. Basic SAR theory, sensors, and algorithms are covered. The computational requirements of some common SAR

algorithms are derived and compared. Finally, the partitioning of SAR processing for parallel implementation is examined.

Section 3 gives some background material that is necessary to set up the framework within which the study of architectures will occur. Among other things, the criteria by which the SAR architectures will be judged are established.

Section 4 gives a high level classification of architectural approaches. This is based on the regular parallel processor classifications, but is adapted to suit SAR processors.

Section 5 identifies the technology areas that are the key drivers behind SAR processor design, and surveys the state-of-the-art in these areas.

The architectures from Section 4 and the technologies from Section 5 are combined to find some possible SAR processor architectures. These possibilities are discussed in **Section 6** and the most promising ones are identified for further study.

Section 7 examines some of the issues related to memory systems design for SAR processors. Memory access patterns during SAR processing are characterized. Various techniques for designing fast memory systems for SAR processors are presented.

Interconnection networks for parallel SAR processors are considered in **Section 8**. The most promising ones are analyzed and compared.

Section 9 performs a detailed examination of some possible SAR processor architecture designs. Their suitability for SAR processing is determined and, where possible, the results of a quantitative analysis are presented.

The conclusions of the thesis are given in **Section 10**.

2 Overview of SAR Processing

This section gives a general overview of the SAR processing problem. The topics addressed include basic SAR theory (Section 2.1), types of SAR sensors (Section 2.2), types of SAR processors (Section 2.3), common SAR processing algorithms (Section 2.4), the computational requirements of SAR algorithms (Section 2.5), and the partitioning of SAR processing for parallel processing (Section 2.6).

2.1 SAR Processing Requirements

This section is a summary of the main elements of SAR processing theory that are useful in deriving the requirements for SAR processor architectures. This information is widely reported in the literature but is not easily usable because it is either presented as part of a larger theoretical treatment of SAR or is given for only a specific algorithm implementation. This section attempts to provide a concise reference to the key equations of SAR processing theory. No attempt is made to give a thorough treatment of SAR imaging. SAR has been described and studied in numerous papers and several books [27][36][34][87]. The discussion in this section is based on [27] and [28].

2.1.1 General SAR Background

The general geometry of a SAR imaging system is shown in Figure 2.1. As the SAR platform travels over the surface of the earth it transmits pulses of radio frequency (RF) energy towards the ground. The pulses are repeated at the pulse repetition frequency (PRF). The SAR antenna is aimed in a direction approximately perpendicular to the direction of motion and downwards at an angle γ from the nadir. γ is called the elevation or nadir angle. The SAR platform travels along its flight path at a velocity v and at an altitude of h. The angle between the center line of the antenna and the direction broadside to the platform's motion is called the squint or look angle, θ. The energy transmitted by the antenna illuminates a footprint on the ground, the area of which is determined by the size of the antenna, the range, and the radar wavelength. As the platform moves, the radar images a strip along the ground called the swath. The radar pulse is scattered by objects on the ground and some of the energy is reflected back towards the radar antenna where it arrives after a time delay proportional to the distance between the radar and target. The range of a target from the radar is $r = t \cdot c/2$, where t is the time variable in the range direction, c is the speed of light and the factor of two accounts for round trip propagation delay. The range between the radar and a target at the time of their closest approach is denoted by r_0. The range between the radar and target when the target passes through the center of the antenna beam is denoted by r_c and is equal to r_0 when $\theta = 0$. The dimension along the radar's path is called the azimuth direction and the azimuth time variable is denoted

by η. This means that the position of the radar in azimuth relative to the origin is equal to $v \cdot \eta$. The time at which the platform is closest to the target is η_0.

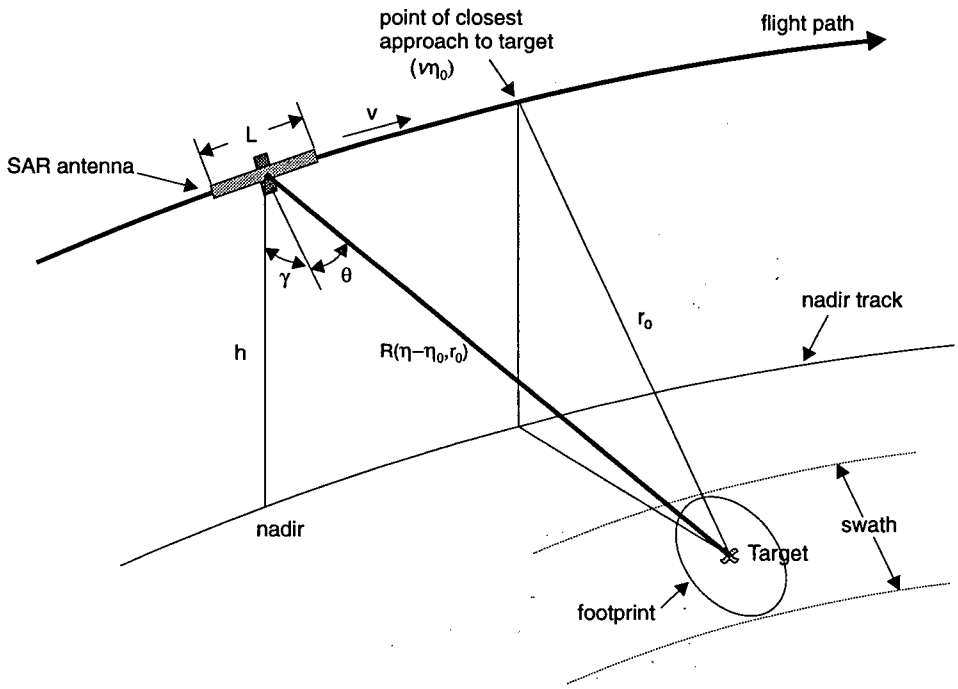


Figure 2.1: General SAR geometry

The received SAR signal is digitized and stored in the memory of the signal processor as a two dimensional array of samples. One dimension of the array represents distance in the slant range direction between the sensor and the target. The other dimension represents the azimuth direction. Each line of data with a common azimuth index represents data collected during one pulse repetition interval.

A number of assumptions are commonly made in analysing SAR processing. It is convenient to consider only the case of a single point scatterer. This is allowable since the SAR system is linear and a complete image is made up of a superposition of point scatterers. In this section rectilinear geometry is assumed, that is, it is assumed the platform is travelling in a straight line.

From Figure 2.1 it can be seen that the instantaneous range to a target with range of closest approach r_0 is:

$$R(\eta - \eta_0, r_0) = \sqrt{r_0^2 + v^2 (\eta - \eta_0)^2}. \tag{2.1}$$

Since the range of a target depends on its position in azimuth, the returned echoes from a target migrate in range as the radar footprint sweeps over the target. The drift of the returns from a target in range is called range cell migration (RCM). The trajectory of a point target response in the received signal memory is

shown in Figure 2.2 [28]. The figure depicts the case of a non-zero squint angle. RCM must usually be accounted for in SAR processing. RCM Correction (RCMC) constitutes one of the main complications in the processing of SAR data. If RCMC were not necessary, then SAR processing would be substantially simplified.

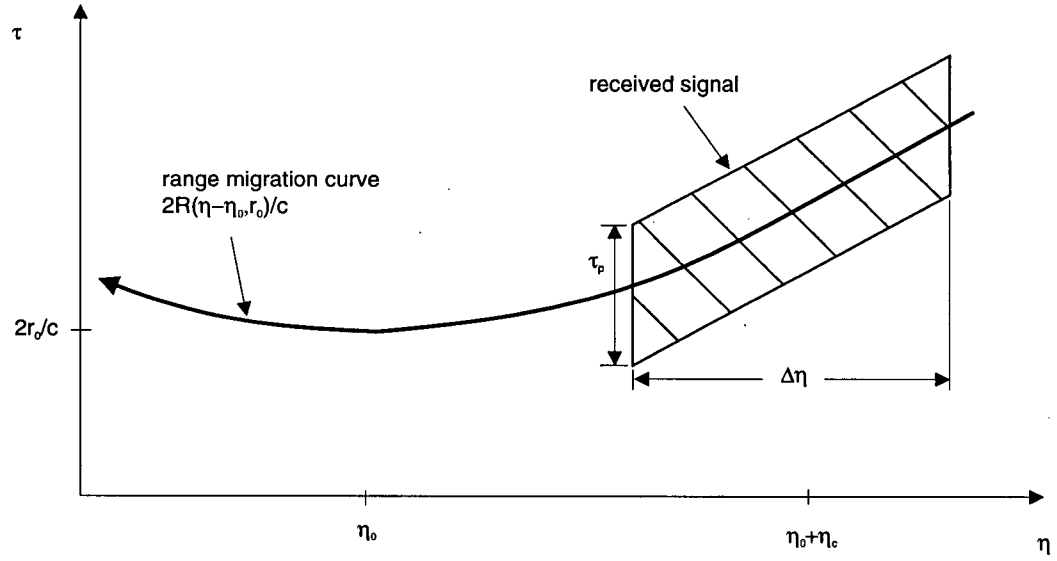


Figure 2.2: Trajectory of point response

The pulse transmitted by the radar can be expressed as

$$s_{tx}(\tau) = Re\left(p(\tau) \cdot e^{j \cdot 2 \cdot \pi \cdot f_0 \cdot \tau} \right), \tag{2.2}$$

where $p(\tau)$ is the complex baseband representation of the pulse, and f_0 is the carrier frequency. The wavelength of the carrier is $\lambda = c/f_0$. Typically a linear frequency modulation (FM) pulse with a rectangular amplitude of duration τ_p is used. An FM pulse or chirp is superior to using a simple rectangular pulse since it allows much greater range resolution to be attained for the same peak transmitter power. An FM chirp can be represented by

$$p(\tau) = rect\left(\frac{\tau}{\tau_p} \right) \cdot e^{-j \cdot \pi \cdot K_r \cdot \tau^2}, \tag{2.3}$$

where K_r is the frequency modulation rate. The bandwidth of the chirp is $\Delta f_\tau = K_r \cdot \tau_p$. The signal received by the antenna from a point scatterer in response to a transmitted pulse is a delayed version of the same pulse. The delay time for the echo to return to the antenna is

$$2 \cdot \frac{R(\eta - \eta_0, r_0)}{c}. \tag{2.4}$$

Using equations 2.2 and 2.4, the received SAR signal from a point scatterer at (η_0, t_0) in complex base-band form can be represented as

$$d(\eta, \tau) = \sigma' \cdot w(\eta - \eta_0 - \eta_c) \cdot p\left(\tau - \frac{2 \cdot R(\eta - \eta_0, r_0)}{c}\right) \cdot e^{\frac{-j \cdot 4 \cdot \pi \cdot f_0 \cdot R(\eta - \eta_0, r_0)}{c}}, \qquad (2.5)$$

where σ' is the scatterer reflectivity and $w(\eta)$ is the antenna pattern in azimuth. The parameter η_c in the antenna pattern is the beam center offset time and represents the effect of the squint angle. The time that the target passes through the beam center is offset from the time of closest approach by η_c as can be seen in Figure 2.2. If the squint angle is zero then η_c is also zero.

The basic intent of SAR processing is to recover a single point target response from the returned signal, which is represented by equation 2.5. This consists of a two-dimensional space variant convolution that compresses the target response of equation 2.5 to a single point. Usually the range and azimuth directions can be looked at separately during processing. In each dimension, the data needs to be convolved with a matched filter. In the range direction, a matched filter compresses the received chirp to a single point response. In azimuth, the situation is complicated by the fact that the returned signal from a single point target appears at a number of ranges during the time that it is illuminated by the radar. All the energy corresponding to a target needs to be collected and then compressed with another matched filter. This filter operates by matching the azimuth variation in phase of the received signal for a point target. This variation is expressed by the appearance of the factor $R(\eta - \eta_0, r_0)$ in the exponent in equation 2.5. Thus, the phase variation can be predicted from the geometry of the situation.

The general theory of SAR processing will not be discussed further in this thesis. The next sections will focus on describing the equations that are used in defining the requirements for a SAR processor.

2.1.2 Azimuth Frequency

A key step in analyzing the SAR processing problem is to examine the frequency content of the received signal in the azimuth direction. If the derivative of the azimuth phase term in the equation for the received signal (Equation 2.5) is taken, an expression for the instantaneous azimuth-frequency can be obtained. By also considering the geometry of the situation, the expression can be written as follows [28].

$$f_\eta = \frac{2 \cdot v \cdot f_0 \cdot \sin(\theta_s(\eta - \eta_0))}{c} \quad \text{or} \quad f_\eta = \frac{2 \cdot v \cdot \sin(\theta_s(\eta - \eta_0))}{\lambda}, \qquad (2.6)$$

where $\theta_s(\eta - \eta_0)$ is defined to be the instantaneous squint angle. At the beam center θ_s equals θ, the nominal squint angle. The signal energy is centered in azimuth frequency at the Doppler centroid:

$$f_{\eta c} = \frac{2 \cdot v \cdot f_0 \cdot \sin(\theta)}{c} \quad \text{or} \quad f_{\eta c} = \frac{2 \cdot v \cdot \sin(\theta)}{\lambda}. \tag{2.7}$$

The azimuth bandwidth for small squint angles is

$$B_D = \frac{2 \cdot v}{L}. \tag{2.8}$$

The amount of Doppler bandwidth (B_D) that is utilized by the SAR processor is called the processed bandwidth (B_p).

For small squint angles, the azimuth frequency to azimuth time relationship can also be expressed as:

$$f_\eta = K_a \cdot (\eta - \eta_0), \tag{2.9}$$

where K_a is the azimuth FM rate as given by the following expression:

$$K_a = -\frac{2 \cdot v^2}{\lambda \cdot r_0}. \tag{2.10}$$

The values of the Doppler centroid and the azimuth FM rate are needed by all SAR processing algorithms. Various methods of calculating and updating these parameters exist for different processing situations.

2.1.3 Resolution

The resolution in the range and azimuth directions are fundamental parameters of every SAR system. In range, the spatial resolution in the slant range plane is

$$\delta r = \frac{c}{2 \cdot \Delta f_\tau} \text{ m}. \tag{2.11}$$

This shows that the range resolution is inversely proportional to the chirp bandwidth.

In azimuth, the spatial resolution on the surface is

$$\delta x = \frac{v}{B_p} \text{ m}, \tag{2.12}$$

which comes from the fact that the azimuth resolution is inversely proportional to the processed azimuth bandwidth. An equivalent formulation (for rectilinear geometry) is:

$$\delta x = \frac{L}{2} \text{ m.} \tag{2.13}$$

This is a key result of SAR processing theory. The azimuth resolution is independent of the distance between antenna and target, and actually improves with a smaller antenna!

2.1.4 Range Cell Migration

The expression for the range of the target from the radar (Equation 2.1) is often approximated by a second order Taylor series approximation [27]:

$$R(\eta) \approx r_c - \frac{\lambda \cdot f_{\eta c}}{2} \cdot (\eta - \eta_c) - \frac{\lambda \cdot K_a}{2} \cdot \frac{1}{2} \cdot (\eta - \eta_c)^2 \tag{2.14}$$

or

$$R(\eta) \approx r_c - v \cdot \sin\theta \cdot (\eta - \eta_c) + \frac{v^2}{2 \cdot r_c} \cdot (\eta - \eta_c)^2. \tag{2.15}$$

The linear term of this approximation is referred to as range walk while the quadratic term is called range curvature.

For zero squint, $\theta = 0$ and $\eta_c = 0$, a simplified equation for the range to the target applies:

$$R(\eta) \approx r_0 + \frac{v^2}{2 \cdot r_0} \cdot (\eta - \eta_0)^2. \tag{2.16}$$

In this case, the range walk component is zero. The amounts of range walk and range cell migration are critical in choosing a processing algorithm and designing a processor. The maximum amount of RCM to be expected in a processing scenario can be determined from the equations given for $R(\eta)$ above. The maximum amount of range walk is

$$N_{RW} = \frac{\lambda \cdot f_s \cdot f_{\eta c}^{max} \cdot B_p}{c \cdot K_a^{min}} \text{ samples} \quad \text{or} \quad N_{RW} = \frac{\lambda \cdot f_{\eta c}^{max} \cdot B_p}{2 \cdot K_a^{min}} \text{ m.} \tag{2.17}$$

One key decision in SAR processor design is whether or not it is necessary to correct for range walk. The amount of range walk is largely determined by the size of the squint angle. Correcting for range walk can

imply an algorithm modification such as secondary range compression (SRC) in the range-Doppler algorithm. The criteria for needing SRC in range-Doppler is

$$\frac{TB}{N_{RW}^2} \leq 1, \tag{2.18}$$

where $TB = B_p \cdot \tau_c$ is the azimuth time bandwidth product and τ_c is the coherent integration time [27].

The maximum amount of range curvature expected is

$$N_{RC} = \frac{\lambda \cdot f_s \cdot B_p^2}{8 \cdot c \cdot K_a^{min}} \text{ samples} \quad \text{or} \quad N_{RC} = \frac{\lambda \cdot B_p^2}{16 \cdot K_a^{min}} \text{ m}. \tag{2.19}$$

By assuming $B_p = B_D$ and substituting Equations 2.8 and 2.10 into Equation 2.19, the amount of range curvature can also be written as:

$$N_{RC} = \frac{\lambda^2 \cdot r_0}{8 \cdot L^2} \text{ m}. \tag{2.20}$$

The amount of range curvature is another factor that determines the type of algorithm that can be used. If range curvature is too large then the processing algorithm must correct for it. The rule of thumb for choosing an algorithm that cannot correct for range curvature (like SPECAN) is that the range curvature be less than one cell. Setting N_{RC} to one in Equation 2.19 results in the following criteria for not requiring range curvature correction:

$$|K_a| \geq \frac{\lambda \cdot f_s \cdot B_p^2}{8 \cdot c} \quad \text{or} \quad TB = |K_a| \cdot \tau_c^2 \leq \frac{8 \cdot c}{\lambda \cdot f_s}. \tag{2.21}$$

It can be seen from Equations 2.18 and 2.21 that the azimuth time bandwidth product is a key factor in determining which algorithm can be used. The time bandwidth product is a fundamental characteristic of a SAR system. Low time bandwidth product systems are typical of high frequency (e.g. X-band) SARs.

2.1.5 Correlation Processing

When the data is processed, the correlation replicas in both range and azimuth have the form:

$$S_r(t) = e^{2 \cdot \pi \cdot j \cdot \frac{K}{2} \cdot t^2}. \tag{2.22}$$

The length of the range reference function is simply the length in samples of the transmitted chirp:

$$L_r = f_s \cdot \tau_p . \tag{2.23}$$

When using fast convolution to convolve the SAR data with the correlation replica, L_r, output samples have to be discarded from each output block due to wrap-around. Thus a range processing efficiency factor can be defined as

$$g_r = \frac{N_r - L_r}{N_r} . \tag{2.24}$$

The range reference function does not have to be updated very often since the properties of the transmitted chirp usually do not vary greatly.

The length of the azimuth reference function is given by

$$L_{az} = \frac{\lambda \cdot r_0 \cdot PRF}{L \cdot v} \cdot \frac{B_p}{B_D} \quad \text{or} \quad L_{az} = \frac{B_p \cdot PRF}{K_a} . \tag{2.25}$$

L_{az} is proportional to the azimuth beamwidth, which is proportional to range. This means that a processor may have to use different azimuth matched filter lengths at different ranges.

Some convenient vector length, N_{az}, needs to be chosen for azimuth processing that is long enough to provide reasonable efficiency and not so long as to require unrealistic amounts of memory. Figure 2.3 illustrates the splitting of the SAR data into azimuth blocks and the overlap regions that are required between blocks. The overlap between blocks must be adjusted according to the azimuth shift of each block relative to the next one. This azimuth shift is

$$N_s = (N_{az} - L_{az}) \cdot \frac{\dot{f}_{\eta c}}{K_a} . \tag{2.26}$$

An azimuth processing efficiency factor can be written as

$$g_a = \frac{N_{az} - L_{az} - N_s}{N_{az}} , \tag{2.27}$$

where N_s is taken to be the maximum block to block shift.

In addition to considering the processing efficiency, the azimuth block length must also take into consideration the requirements for updating the azimuth reference function in the along-track direction. The update

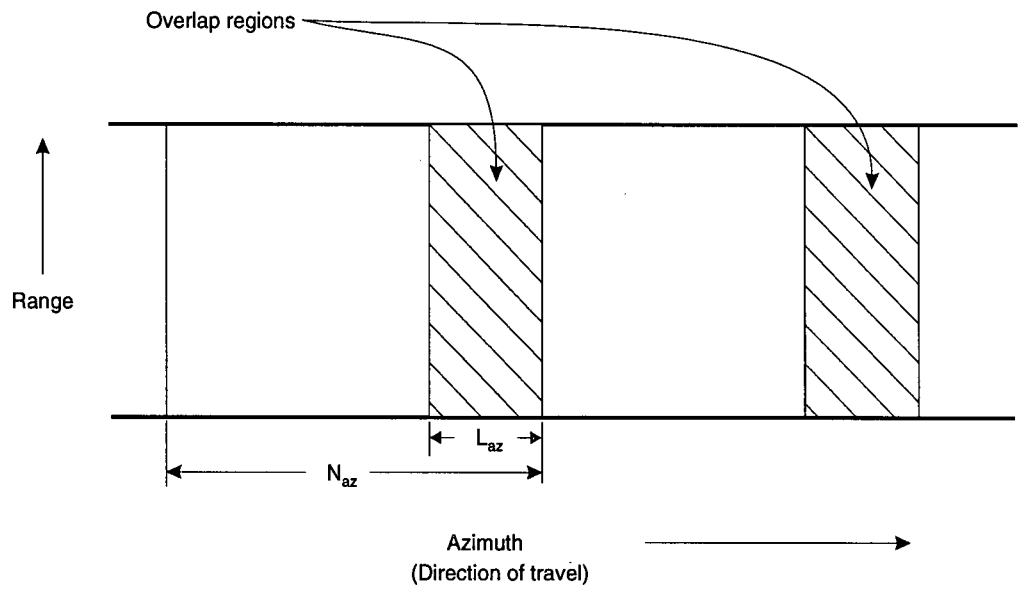


Figure 2.3: Block processing of SAR data

interval depends on the Doppler drift rates, the rates of change of K_a and $f_{\eta c}$, and the amount of error that is allowable. The reference function update interval sets an upper limit on the azimuth block size N_{az}, although this is very rarely a restriction in practice.

The azimuth reference function must also be updated in the range direction. The update interval in range is driven by the slant range dependence of K_a. In order to keep errors resulting from incorrect K_a from causing problems, the number of azimuth lines between updates is usually on the order of one to eight depending on the error specification.

2.1.6 Data Set Sizes

The choice of the data set size in the range direction is relatively straightforward. The range extent of the received signal is determined by the antenna pattern in range. The 3 dB beamwidth of the antenna is $\theta_v = \lambda/H$ radians, where H is the height of the antenna. If R_m is the range at mid-swath, then the extent of the radar beam in slant range is

$$W_r = \frac{\lambda}{H} \cdot R_m \quad \text{m}. \tag{2.28}$$

Slant range, SR, is related to ground range, GR, by

$$SR = \frac{GR}{\sin\gamma} \text{ m}. \tag{2.29}$$

The conversion between slant range in meters, SR, and number of range samples, N, is

$$N = \frac{2 \cdot f_s}{c} \cdot SR \quad \text{samples}. \tag{2.30}$$

The size of the data array in range is also bounded by the pulse repetition period $T_p = 1/(PRF)$. If the sampling rate in range is f_s, then the maximum number of range samples is $T_p \cdot f_s$. The range sampling rate is chosen to be large enough to unambiguously sample the signal in range which has a bandwidth equal to the transmitted pulse bandwidth of $\Delta f_\tau = \tau_p \cdot K_r$.

Thus the constraints on N_r, when processing the full range swath, are

$$\frac{\lambda}{H} \cdot R_m \cdot \frac{2 \cdot f_s}{c} < N_r < T_p \cdot f_s. \tag{2.31}$$

Typical values of N_r are 2000 to 10000 samples. Swath widths smaller than the full range swath can also be processed. This simply results in an image that covers a smaller extent in range with no loss of resolution.

In azimuth, the situation is significantly more complicated. The antenna 3 dB beamwidth in azimuth is $\theta_v = \lambda/L$ radians where L is the length of the antenna. This means that the exposure time for small squint angles is

$$\Delta\eta = \frac{\lambda \cdot r_0}{L \cdot v} \quad \text{sec}. \tag{2.32}$$

The size of the data array that spans the full exposure time is

$$N_{az} = \Delta\eta \cdot PRF = \frac{\lambda \cdot r_0}{L \cdot v} \cdot PRF \quad \text{samples}. \tag{2.33}$$

There are many other constraints that affect the choice of N_{az}. As mentioned in the previous section, the reference function update interval can impose an upper bound on N_{az}. The desired processing efficiency (Equation 2.27) imposes a lower limit on N_{az}. If a value of N_{az} smaller than that called for in Equation 2.33 is used, the processed azimuth bandwidth is reduced and the azimuth resolution is degraded. If a larger value of N_{az} is used, the azimuth resolution does not improve significantly. N_{az} is often chosen to be a power of two to accommodate fast Fourier transform (FFT) algorithms. Some typical values are 2048, 4096 and 8192.

2.1.7 Summary

The preceding sections have given a brief summary of some basic equations related to SAR processing. This material gives the necessary background for the comparison of SAR algorithms and the derivation of their processing requirements in later sections. Additional information on the topics touched on here can be found in [27]. The next section gives some examples of SAR sensors, and uses the parameters of those sensors to illustrate the use of the equations given in this section.

2.2 SAR Sensors

SAR sensors fall into two general categories: airborne and spaceborne. Although the theory for both types of sensors is similar, there are some differences in the data processing. This section lists some of the key characteristics of selected air- and spaceborne SAR sensors. This information was drawn together and presented here since it is quite difficult to find detailed parameter lists of SAR sensors in the literature.

A number of SAR sensors have been flown in aircraft by several countries. Table 2.1 gives a brief summary of some of the parameters for a few airborne SARs [27][67].

Table 2.1: Parameters of selected airborne radars

System	NASA/JPL	CCRS/MDA	China/MDA IRIS X2C
Aircraft	DC-8	CV-580	Lear-35A
Frequency (GHz)	0.44,1.25,5.3	5.3,9.25	9.6
Polarization	Quad	Dual (like, cross)	H
Az resolution (m) / Looks	8 / 4	6 / 7	3 / 4 or 6 / 8
Range bandwidth (MHz)	20,40	35	60
Swath width (km)	10-18	18-65	10,20
Elevation angle γ (deg)	10-65	0-87	not available
Quantization (bits)	8	6	8

Airborne SARs typically offer higher resolution than satellite SARs but cover a smaller area per unit time. In airborne systems, the processing is simplified by the fact that RCM is not as great as for space-borne SARs, but is complicated by other issues such as motion compensation and greater variation in range of processing parameters.

The first SAR equipped satellite was Seasat which was launched in 1978. Although its mission lasted only about 100 days, it collected a vast amount of data that was analyzed for many years. The success of Seasat lead to several SAR missions on the space shuttle (SIR-A, B, C series). Over ten years passed before another SAR satellite was launched. However, now several SAR satellites are in orbit and several more are

planned. The next several years will see more SAR data becoming available than ever before. Table 2.2 summarizes the properties of several spaceborne SARs [27][77][72].

Table 2.2: Parameters of selected SAR satellites

Satellite	Seasat	ERS-1/2	JERS-1	Radarsat	ENVISAT
Primary Parameters (approx.)					
Frequency Band (GHz)	L (1.3)	C (5.3)	L (1.3)	C (5.3)	C (5.331)
Wavelength λ (m)	0.235	0.057	0.235	0.057	0.05627
Polarization	HH	VV	HH	HH	HH, VV, HV, or VH
Altitude (km)	800	785	568	793-821	786-814
Effective Velocity v (m/s)	7100	6928	7500	7071	6800
Nominal slant range (km)	850	850	690	844-1277	830-1130
Antenna Size $L \times H$ (m × m)	10.7 × 2.2	10.0 × 1.0	11.9 × 2.2	15.0 × 1.5	10.0 × 1.3
Transmitted pulse length τ_p (μs)	34	37.1	35	42	11.6-41.3
Range freq. rate K_r (MHz/μsec)	0.53	0.42	0.43	0.27,0.41, 0.71	0.03-1.4
Range sampling rate f_s (MHz)	45.52 (real)	18.96	17.076	12.9-32.2	19.208
PRF (Hz)	1647	1650	1506-1606	1270-1390	1580-2150
Slant range swath (km)	40	39	43	~25-250	~25-250
Ground range swath (km)	100	100	75	50-500	50-500
Elevation angle γ (deg)	23	23	35	20-50	15-45
Expected squint θ (deg)	4	0	4	4	0
Quantization (bits)	analog	5	3	4 (BFP)	4 (BFP)
Downlink rate (Mbps)	analog	105	60	105 or 85	~100
Derived Parameters (approx.)					
Azimuth resolution (m) / Looks	23 / 4	25 / 3	30 / 4	28 / 4	30 / 4
Range Bandwidth (MHz)	19	13.5	15	11.6,17.3,30	0.4-16
Azimuth FM Rate K_a (Hz/s)	500	2340	690	1350-2100	1450-2000
Doppler Bandwidth B_D (Hz)	1370	1385	1260	940	1350
Range matched filter length L_r	774	705		542-1352	223-793
Az matched filter length L_{az}	4330	976	2930	622-970	1450-1900
Min. value of N_a needed to process full exposure time	4330	1154	2918	954	1800
Max. amount of range curvature (m)	51	3.5	34	2.3	4.5

As can be seen from the table, the satellites have similar SAR system parameters in many cases. A major difference arises between the L-band satellites (Seasat and JERS-1) and the C-band satellites (ERS-1, Radarsat, and ENVISAT). Larger processor data arrays in azimuth direction are required to process the full target exposure time for the L-band satellites. The L-band systems also exhibit much larger amounts of range curvature, which must be properly handled by the processing algorithm in order to achieve high accuracy.

The newer SAR satellites, like Radarsat and ENVISAT, have a much wider range of operating modes and a larger number of variable parameters. Radarsat has a steerable antenna beam that allows a range of swaths to be imaged. ENVISAT will add an alternating polarization mode to these capabilities. As a result of these multiple modes of operation, the values shown in Table 2.2 do not capture all the possibilities and should be seen as representative values.

2.3 SAR Processors

Until about 1978, all satellite SAR processors used optical techniques. In the following three years, digital processing techniques almost completely replaced the optical ones. Digital processing offers major advantages in terms of accuracy and flexibility. However even today, there are occasions when size, weight or power constraints are crucial for which optical processing is useful. Modern optical processors are quite different from the ones used in the 1970s [45].

It is difficult to generalize a set of requirements for SAR processors since they have such a wide variety of uses. Table 2.3 lists some typical types of SAR processors and some typical attributes for each type [30].

Table 2.3: Attributes of some typical SAR processors

Type	Example SAR Processor	Resolution	Processing rate	Required Processor Speed	Primary Design Constraints
Image Analysis/Research	Radarsat Precision Processor (RPP)	med. to very high	low	varies (typically that achievable on a single workstation)	flexibility
Satellite Quicklook	Radarsat Fastscan	low e.g. 100-180 m	high (RT)	40 MOP/s (at least)	speed
Airborne	MDA IRIS X2C	high e.g. 1 to 5 m	high (RT)	500 MOP/s (at least)	speed, size, accuracy
Satellite	Radarsat SAR Processor (RSARP)	high e.g. 30 m	high (RT) (or some fraction of)	3-4 GOP/s (or some fraction of)	accuracy, speed

Image analysis or research processors are typically used by researchers or scientists in labs for testing algorithm modifications or for verifying the results of production processors. Quicklook processors are typically used for real-time processing of SAR data from a satellite as it is received in order to obtain a quick estimate of image quality and the overall correct operation of the system. Airborne processors are often mounted in the aircraft carrying the SAR sensor, and are used to obtain high quality images in real-time. Satellite SAR processors typically have the most demanding requirements. These processors are usually located at the satellite ground stations and are required to process all the data that the satellite generates at high levels of accuracy. Due to the extremely high processor speed required, most satellite SAR processors are built to achieve some fraction of real-time, for example, one tenth real-time. This is appropriate since the satellite is only in view of the ground station for a portion of the orbit, and full real-time processing is not required to process all the data acquired from the satellite. This limitation is also inherent in the design of the satellite which usually only has enough power to scan for a portion of its orbit.

There are other types of SAR processors that have not been considered here. For example, spaceborne processors have not yet been built but are likely to appear in the future. Such systems will have extreme size, power, weight, and radiation immunity constraints.

2.4 Common SAR Algorithms

This section briefly surveys a representative set of SAR processing algorithms. The purpose of this section is to introduce each algorithm's sequence of operations in preparation for the computational analysis in Section 2.5. The following common SAR processing algorithms or classes of algorithms are considered:

- Time Domain algorithm
- Spectral Analysis (SPECAN) algorithm
- Range Doppler algorithm
- 2-D Frequency Domain algorithms
- Chirp Scaling algorithm

These were chosen because they are either in common use or because they have been shown to be useful for Radarsat processing [65]. The main steps in each of these algorithms are summarized in Figure 2.4. This section characterizes each algorithm in terms of its major processing steps. Some indication of the accuracy of each algorithm is given but is not considered in detail. There are many other SAR algorithms that are not considered here. Many of these are simply variations of the ones discussed in this section. The algorithm descriptions in this section are based on [27], [65], [8], and [28].

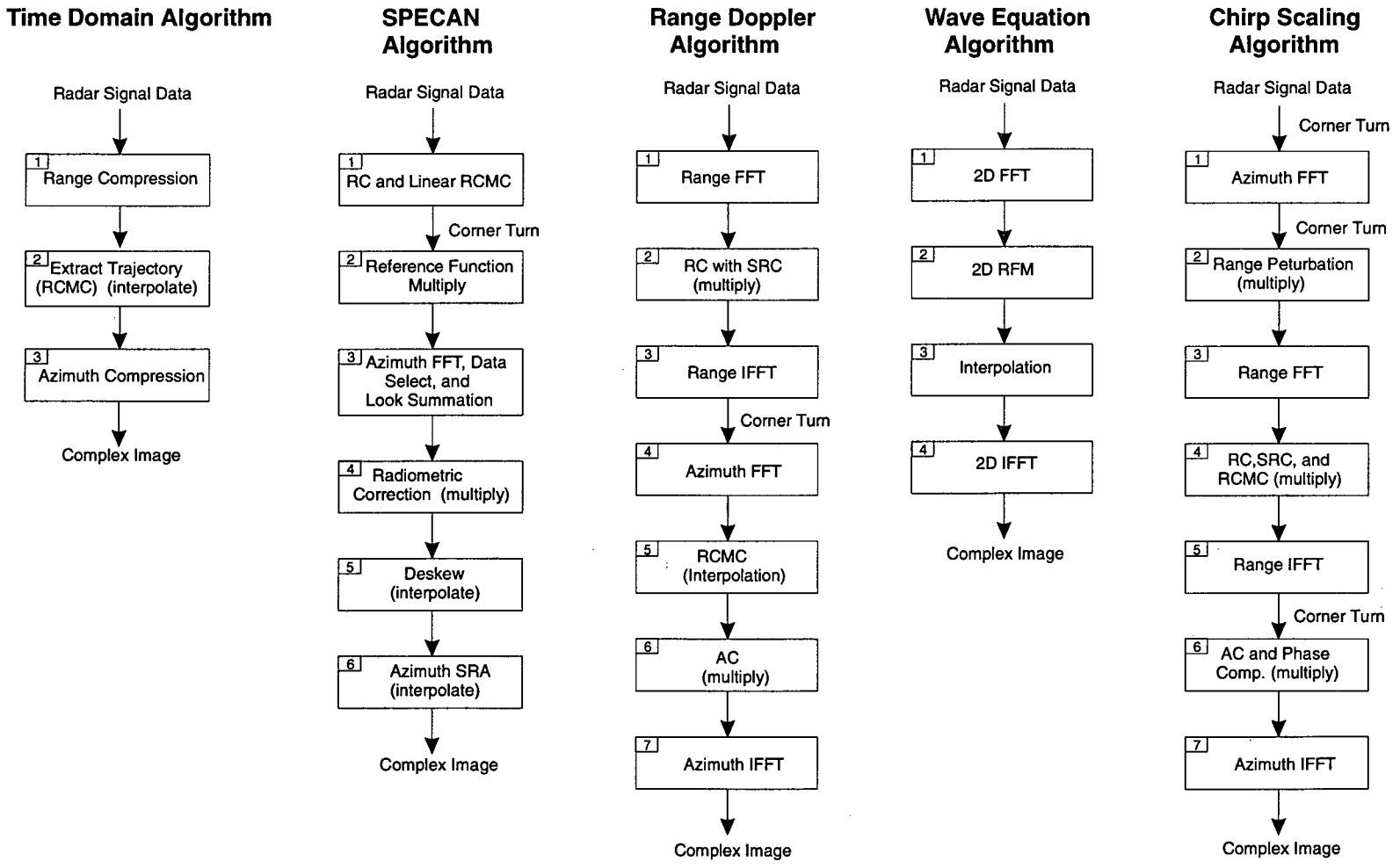


Figure 2.4: Main processing steps in some common SAR algorithms

2.4.1 Time Domain Algorithm

The time domain algorithm is conceptually the simplest SAR algorithm. It directly implements the two-dimensional correlation that is at the heart of SAR processing. The steps in the algorithm are as follows:

Range Compression

The radar signal data is convolved in the range direction with the range matched filter.

Trajectory Extraction

A curved trajectory corresponding to the path of a target through signal memory is extracted by interpolation for each output point.

Azimuth Compression

The data along each trajectory extracted in the previous step is convolved with the azimuth matched filter.

The main advantage of the time domain algorithm is that it can be the most accurate of all algorithms. Parameter variations, as well as any degree of antenna squint, can be handled. Another advantage is that any output sample grid can be chosen, which eliminates the need for later resampling. The main disadvantage of the time domain algorithm is that it is extremely computationally expensive. Time domain convolutions are required in both directions which require large amounts of computations for longer filter lengths. Additionally, for each output point, a complete trajectory must be extracted by interpolation from the data array before being compressed in azimuth.

The huge number of operations required by this algorithm are difficult to achieve in a practical processor. The time domain algorithm is easier to use in applications that do not require RCMC. As a result, time domain approaches have been used in several airborne SAR processors. The development of extremely fast digital filter integrated circuits (ICs) has opened up the possibility of designing hardware implementations of time domain algorithms with reasonable performance. This approach to SAR processor implementation will be examined in a later section.

2.4.2 SPECAN Algorithm

SPECAN is based on the idea of demodulating the returned signals with a linear FM signal which separates the returns from different targets to different frequencies. The targets can then be extracted using a single FFT. SPECAN is more fully described in [63], [79], and [47]. Since SPECAN is essentially a pulse compression technique, it can be used for either range compression, azimuth compression, or both. This section concentrates on the use of SPECAN for azimuth compression.

The steps in the algorithm are as follows:

Range Compression (RC) and Linear RCMC

Range compression is performed in any convenient manner, for example, either by fast convolution or by SPECAN. Then a skew or interpolation is done in the range direction to compensate for range walk.

Reference Function Multiply

The data is multiplied by a linear FM waveform with an FM rate equal to the negative of the azimuth FM rate, K_a, in the azimuth direction. The reference function is updated as a function of cross-track position to track the Doppler parameter variation.

FFT and Data Selection

Short-length FFT operations are then applied to the data. This has the effect of compressing the returns from a target, which were confined to a single frequency by the previous step, into one bin. The output points of the FFT that correspond to valid data are selected, and the location of the next FFT is chosen.

Radiometric Correction

Radiometric corrections are needed because the look positions depend on azimuth time. This is usually a vector multiply operation.

Deskew

Deskewing is necessary because of the shift in range from the initial linear RCMC. This consists of an interpolation operation.

Azimuth Sample Rate Adjustment

Sample rate adjustment is needed to achieve a uniform output grid if different FM rates are used across the azimuth block. This is a resampling from the range-Doppler grid (fan-shaped) to an orthogonal grid. It consists of an interpolation operation.

The output data after the FFT is not valid for all positions within the processed block. Values near the edges of the block are degraded and must be discarded. This decreases the processing efficiency. The fraction of data that is discarded increases as the required resolutions increases. Improved efficiency can be obtained by shortening the FFT length, but this comes with the penalty of decreased resolution.

Some of the advantages of SPECAN are listed below:

- SPECAN is the most efficient of all algorithms for satellite SAR in terms of both arithmetic and memory.

- SPECAN can efficiently handle short vector lengths which can be important for special scanning modes like Radarsat's ScanSAR mode.

The disadvantages of SPECAN are as follows:

- SPECAN can only achieve about two-thirds of full resolution and still be computationally efficient.
- Quadratic RCMC cannot easily be accommodated.
- Extra processing is required to correct for radiometric scalloping that is present in the output image.
- The output sample spacing is variable and depends on the FM rate of the azimuth matched filter, the PRF, and the FFT length. This complicates processing and necessitates a resampling operation.

2.4.3 Range-Doppler Algorithm

The range-Doppler algorithm is probably the most widely used SAR algorithm. It was first developed by MacDonald Dettwiler and Associates (MDA) and the Jet Propulsion Lab (JPL) in 1979 for the processing of Seasat data [26][93]. Since then, numerous variations and improvements have been developed, the most important of which is the addition of SRC [53]. Range-Doppler was designed to offer block processing efficiency in both the range and azimuth directions. Range curvature is corrected in the range-time / azimuth-frequency domain which allows for block processing. SRC is a modification applied to the range compression matched filter to correct for range walk and other effects.

The core steps in the range-Doppler algorithm are as follows:

Range FFT

An FFT is performed on the data in the range direction.

Range Compression (RC) with SRC

Range compression and SRC are performed by means of a vector multiply in the range direction.

Range IFFT

An IFFT is performed in the range direction.

Azimuth FFT

An FFT is performed in the azimuth direction.

RCMC

RCMC is performed by a shift and interpolation operation that lines up the target trajectories in memory.

Azimuth Compression (AC)

Azimuth compression is accomplished with a vector multiply in the azimuth direction.

Azimuth IFFT

An IFFT in the azimuth direction completes the formation of the image.

Some of the advantages of the range-Doppler algorithm are as follows:

- Block processing efficiency is achieved for both RCMC and azimuth correlation.
- Range variations of parameters can be accommodated with ease.

Some of the disadvantages of range-Doppler are listed below:

- There are problems with handling squint that are partially but not completely addressed by SRC.
- Small errors are left by the fact the SRC is usually not made fully range and azimuth variant.
- Interpolation is required to perform RCMC. This is complicated to implement correctly, and usually simple interpolators are used which introduce small errors.

2.4.4 Two Dimensional Frequency Domain Algorithms

This is a class of algorithms that uses 2-D FFTs in their processing. Included in this class are the wave equation (WE) algorithm and its variations [21].

The steps in a typical algorithm of this class are as follows:

2-D FFT

The data is transformed into the two dimensional frequency domain by a 2-D FFT.

2-D Reference Function Multiply (RFM)

The data is then multiplied by a 2-D reference function.

Change of variables or Stolt mapping

This is an interpolation operation that accounts for the space variance of the correlation.

2-D IFFT

The image formation process is completed by a 2-D IFFT.

Some advantages of 2-D frequency domain algorithms are as follows:

- Very accurate processing is possible for certain geometries.
- Squinted SARs can be handled.

Some disadvantages of 2-D frequency domain algorithms are as follows:

- The algorithms usually require interpolations.
- Large variations in the Doppler centroid are difficult to accommodate.

2.4.5 Chirp Scaling Algorithm

The chirp-scaling algorithm is a relatively new algorithm that attempts to combine the advantages of the range-Doppler and wave-equation algorithms [78]. The main features of the algorithm are its high accuracy, the lack of any interpolation operations, and its ability (with some modifications) to handle high squint [28].

The steps in the algorithm are as follows:

Azimuth FFT

The data is transformed into the range-Doppler domain with an FFT in the azimuth direction.

Range Perturbation

The data is then multiplied by a function in the range direction. This is the chirp scaling phase multiply that serves to equalize the range migration term of each scatterer's phase function with one at a certain reference range.

Range FFT

The data is transformed into the two dimensional frequency domain with an FFT in the range direction.

RC, SRC, and RCMC

RC, SRC, and RCMC is accomplished with a vector multiply in the range direction.

Range IFFT

An IFFT in the range direction is then applied to the data.

Azimuth Compression (AC) and Phase Compensation

Azimuth compression and phase compensation is performed with a vector multiply in the azimuth direction.

Azimuth IFFT

The image formation is completed with an IFFT in the azimuth direction.

Some of the advantages of the chirp scaling algorithm are as follows:

- No interpolations are required.

- High squint can be handled.

Some disadvantages of the algorithm are listed below:

- Reference function calculations are more involved than in other algorithms.
- It is difficult to handle large variations in the Doppler centroid.
- Chirp scaling is not as well established as other older algorithms.

2.4.6 Algorithm Variations

There are many variations and options that can be applied to the basic SAR image formation algorithms described in the previous sections. The processing of multiple looks is a common algorithmic enhancement. Multiple looks are used to reduce the amount of speckle that is a characteristic of SAR images in exchange for a loss in resolution. Multi-look processing consists of splitting the azimuth bandwidth and processing multiple independent views of the same area and then incoherently summing them.

The algorithm descriptions given so far have only covered the basic image formation operations. There are many post-processing operations that can be applied to the outputs of any of the basic algorithms. These post-processing operations can form a significant portion of the computational requirements of a SAR processor. The exact nature of the post-processing operations applied to SAR images depends heavily on the final application. Some operations that may be performed are radiometric corrections and geometric corrections like resampling and geocoding. Radiometric and geometric calibrations are described in Sections 7 and 8 of [27]. A sample sequence of post-processing operations is shown in Figure 2.5.

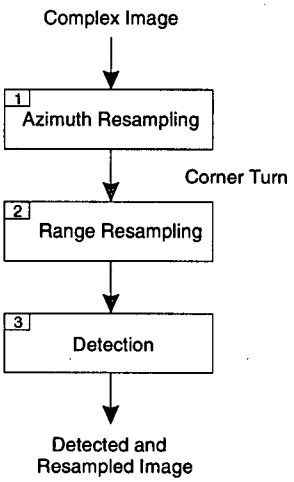


Figure 2.5: Sample post-processing sequence

2.5 Computational Analysis of SAR Algorithms

This section analyzes the computational requirements of the basic SAR processing algorithms presented in the last section. The results presented here extend those previously reported to a wider range of algorithms. A method of counting computing operations based on fundamental vector operations is given (Section 2.5.1). Expressions for the number of operations required by each step in each algorithm are derived (Section 2.5.2 and 2.5.3). Sample results are plotted for common processing situations. Finally a more detailed computational model of the range-Doppler algorithm is presented as a basis for architectural mappings in later sections (Section 2.5.4).

2.5.1 Operation Types

This section describes the types of operations typically needed in SAR processing and the number of operations (OPs) required for each operation on a vector of length N. To calculate computation amounts, a single operation (OP) on real operands is used as the base unit. An OP is defined as either an addition, subtraction, multiplication, or division. If floating point arithmetic is used in the processor, then the OPs can be interpreted as floating point operations (FLOPs).

One dimensional FFT/IFFT

The usual radix two algorithm is assumed with $N/2 \cdot \log_2 N$ butterflies, 10 OPs per butterfly (four multiplies and six adds), and a total of $10 \cdot N/2 \cdot \log_2 N$ OPs.

Complex vector, element-by-element multiplication

Each complex multiplication consists of four real multiplications and two adds, or $6 \cdot N$ OPs.

Complex vector / real vector, element-by-element multiplication

Each complex/real multiplication consists of two real multiplications, or $2 \cdot N$ OPs.

Real vector scalar operations

Each sample requires one operation, or N OPs.

Filtering with Complex Data and Complex Coefficients

Each filtered sample requires $4 \cdot N_{filter}$ OPs of multiplication and $4 \cdot N_{filter} - 2$ OPs of addition for a total of $8 \cdot N_{filter} - 2$ OPs.

Filtering with Complex Data and Real Coefficients

Each filtered sample requires $2 \cdot N_{filter}$ OPs of multiplication and $2 \cdot (N_{filter} - 1)$ OPs of addition for a total of $2 \cdot (2 \cdot N_{filter} - 1)$ OPs. This type of filtering commonly occurs in SAR processing as an interpolation operation.

Filtering with Real Data and Real Coefficients

Each filtered sample requires N_{filter} OPs of multiplication and $(N_{filter}-1)$ OPs of addition for a total of $2 \cdot N_{filter}-1$ OPs.

Rounding functions

Rounding functions are assumed to require 5 OPs.

Root functions

Square root functions are assumed to require 10 OPs.

The operation types and their operation counts are summarized in Table 2.4.

Table 2.4: Operation types

Operation Type	Symbol	Number of Operations
One dimensional FFT/IFFT	$FFT1d(N)$	$10 \cdot N/2 \cdot \log_2 N$
Complex vector element -by-element multiplication	$CVM(N)$	$6 \cdot N$
Complex vector / real vector element-by-element multiplication	$CVRVM(N)$	$2 \cdot N$
Real vector scalar operations	$RVS(N)$	N
Filtering with complex data and complex coefficients	$FilterCC(N_{filter})$	$8 \cdot N_{filter}-2$
Filtering with complex data and real coefficients	$FilterCR(N_{filter})$	$2 \cdot (2 \cdot N_{filter}-1)$
Filtering with real data and real coefficients	$FilterRR(N_{filter})$	$2 \cdot N_{filter}-1$
Rounding	$Round$	5
Roots	$Root$	10

2.5.2 Computational Requirements of SAR Algorithms

This section analyzes the computational requirements of the algorithms introduced in Section 2.4. The operation types and counts described in the previous section are used to quantify the analysis.

The SAR algorithms are used to process an array of data to form a single look complex image. The only exception is the SPECAN algorithm which is modelled as processing a four look image. Both range and azimuth compression are included. This analysis is an extension of that in Section 9 of [27]. This section covers a more complete range of algorithms and counts the number of operations for an entire image array instead of per input sample. Only azimuth compression was included in the comparison in [27] and thus that comparison was able to show the trade-offs between the azimuth compression algorithms alone. Range compression is included in the analysis of all algorithms in this section since, in some algorithms,

range compression cannot be separated as a distinct step. In some cases, range compression can done by a number of methods and is not restricted to the same method that is used for azimuth compression. In this section, it is assumed that range compression is performed by the same method as azimuth compression. This means that in the analysis of the SPECAN algorithm, range compression is performed using SPECAN. In range-Doppler, range compression is performed by fast convolution.

The key parameters in the analysis are N_r and N_{az}, the sizes of the data array in range and azimuth, and L_r and L_{az}, the lengths of the reference functions in range and azimuth. The data set size is varied during the analysis. The variation in the computational requirements as a function of the reference function lengths is examined. The reference function lengths are allowed to increase by powers of two from 2^0 to 2^{14}. It is assumed that the lengths of the range and azimuth reference functions are normally equal. In order to keep the scenarios a little more realistic, L_r is not allowed to grow any larger than 1024. Range reference functions are not usually longer than this. The data set sizes are kept consistent with the reference function lengths by always setting the dimensions of the data array to be twice the azimuth reference function length: i.e. $N_r = 2 \cdot L_{az}$ and $N_{az} = 2 \cdot L_{az}$. This arrangement results in processing situations that are somewhat artificial since in practice reference function lengths and array sizes cannot be chosen arbitrarily. However, the arrangement is useful to show how the amount of computation grows as the problem size increases. It is assumed all data values are complex numbers.

Some of the other definitions and assumptions used in the algorithm models are defined below.

- The data array sizes are assumed to be constant throughout the processing. This ignores changes in the data array dimensions due to filter throw-away regions, which can have a quite significant effect on data array sizes and, therefore, on the amount of computations. This assumption also implies that the azimuth processing step operates on the full amount of data output by the range processing step. Often only a portion of this data will need to be compressed in azimuth since the size of the data array was probably padded in the range direction in preparation for range compression. Since the purpose of this section is only to get rough estimates and compare the algorithms, these issues are not a big concern.
- The analysis adds the updating of the azimuth reference function to the algorithm steps shown in Figure 2.4. The azimuth reference function is assumed to be updated every n_u azimuth lines. n_u is taken to be four in this section. The number of operations required to calculate each value in an azimuth matched filter is OP_{ref} which is taken to be equal to 10 operations.

- Any operations that are not part of the core SAR algorithm or have much smaller computation requirements than the major steps are ignored. This includes any preprocessing operations, calculations of range reference functions and SRC coefficients, Doppler centroid calculations, calculation of RCM coefficients, and any post-processing operations like resampling or detection.
- The overlap between azimuth processing blocks is taken into account with the azimuth processing efficiency factor g_a. g_a is given by equation 2.27 with N_s taken to be zero. It is assumed that range processing is done in a single block so the range processing efficiency is one. The effect of multi-block range processing is taken up in Section 2.6.4.
- The length of the filters used in the interpolation operations required by several of the algorithms is represented by N_{filter} and is taken to be equal to four. The PRF is assumed to be 1500 Hz.

The analysis of each algorithm except SPECAN assumes a single look product. This is justified because processing a single look product requires more computation than a multi-look product. The difference in the amount of computation in the case of the range-Doppler algorithm arises out of the fact that the multi-look case has multiple shorter azimuth FFTs and requires look summation. The single and multi-look cases can be compared by looking at the amount of computation required by the inverse azimuth FFT and look summation. In the single look case, the azimuth IFFT requires $FFT1d(N_{az})$ operations. In the multi-look case the number of operations is

$$N_{look} \cdot FFT1d\left(\frac{N_{az}}{N_{look}}\right) + N_{az} \tag{2.34}$$

which can be written as:

$$FFT1d(NAZ) - NAZ \cdot \left(5 \cdot \log_2(N_{look}) - 1\right). \tag{2.35}$$

This shows that if the number of looks is greater than two, the amount of computation is smaller for the multi-look case. This means that the single look case should be used when deriving worst case computation requirements. Multi-look processing may, however, require more memory.

For each algorithm, bar graphs are shown that depict the relative numbers of operations that are performed in each step of the algorithm and the relative number of operations that are due to each of the operation types of Section 2.5.1. These results are given for a reference function length of 2^{10}: i.e. $L_{az} = 1024$ and $L_r = 1024$.

Time Domain Algorithm

The number of operations required by each step in the time domain algorithm is shown in Table 2.5.

Table 2.5: Computational analysis of time domain algorithm

No.	Step in Time Domain Algorithm	Number of Operations
1	Range Compression	$N_r \cdot N_{az} \cdot FilterCC(L_r)$
2	RCMC (interpolate)	$L_{az} \cdot N_r \cdot N_{az} \cdot FilterCR(N_{filter})$
3	Azimuth Compression	$N_r \cdot N_{az} \cdot FilterCC(L_{az})$
ref	Reference function generation	$\frac{N_r}{n_u} \cdot (OP_{ref} \cdot L_{az})$

Figure 2.6 shows a graph of the relative number of operations in each step of the algorithm. The adjoining graph shows the number of operations that are due to each of the basic operation types.

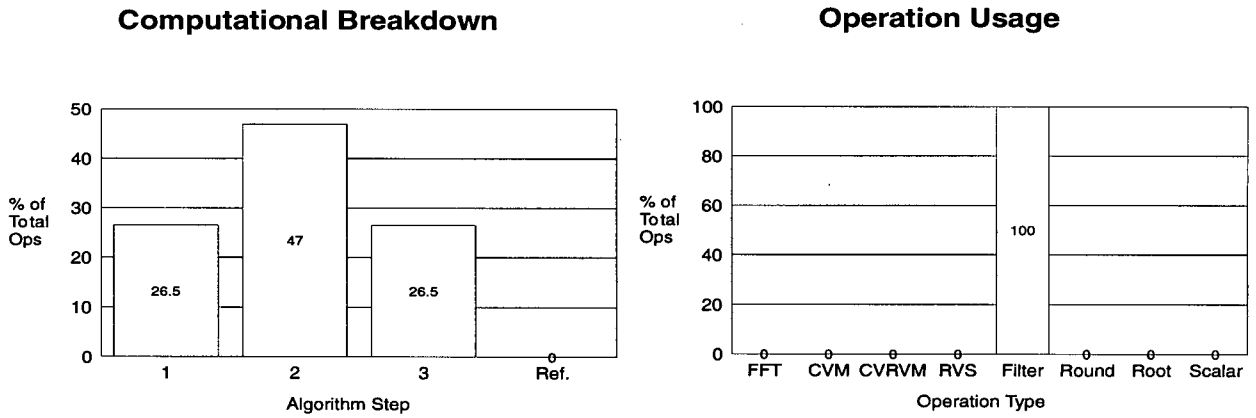


Figure 2.6: Computational analysis of time-domain algorithm

SPECAN Algorithm

Table 2.6 shows the operations required by each step in the SPECAN algorithm. SPECAN is handled somewhat differently than the other algorithms. This is due to the fact that SPECAN cannot be used to process apertures as long as the other algorithms. SPECAN is modelled as processing an aperture that is only 1/4 as long. In order to make the data sizes handled by all the algorithms the same, it is assumed that SPECAN is used to process four azimuth looks. This is a realistic assumption since SPECAN becomes more efficient as the resolution decreases. In Table 2.6 the steps involved in range compression and the look summation step were added to the basic algorithm steps in Figure 2.4. A much more complete discussion of the computational requirements of SPECAN can be found in [79].

Table 2.6: Computational analysis of SPECAN algorithm

No.	Step in SPECAN algorithm	Number of Operations
1a	Range reference function multiply	$N_{looks} \cdot N_{az} \cdot CVM(N_r)$
1b	Linear RCMC (interpolate)	$N_{looks} \cdot N_r \cdot N_{az} \cdot FilterCR(N_{filter})$
1c	Range FFT	$N_{looks} \cdot N_{az} \cdot FFT1d(N_r)$
2	Azimuth reference function multiply	$N_{looks} \cdot N_r \cdot CVM(N_{az})$
3	Azimuth FFT and Data Select	$N_{looks} \cdot \frac{N_{looks}}{N_{looks}-1} \cdot N_r \cdot FFT1d(N_{az})$
3a	Look Summation	$N_r \cdot N_{az} \cdot 2$
4	Radiometric Correction (multiply)	$N_r \cdot CVM(N_{az})$
5	Deskew (interpolate)	$N_r \cdot N_{az} \cdot FilterCR(N_{filter})$
6	Azimuth SRA (interpolate)	$2 \cdot N_r \cdot N_{az} \cdot FilterCR(N_{filter})$
ref	Calculate reference function	$\frac{N_r}{n_u} \cdot OP_{ref} \cdot N_{az}$

Figure 2.7 shows the relative number of operations required by each step in the algorithm and by each operation type. The amount of computation for SPECAN is somewhat overestimated in comparison with the other algorithms. Some radiometric and geometric calibration operations that are required by SPECAN are included in the analysis. These types of operations are not included in the analyses of the other algorithms even though the overall system requirements may make them necessary for those algorithms as well.

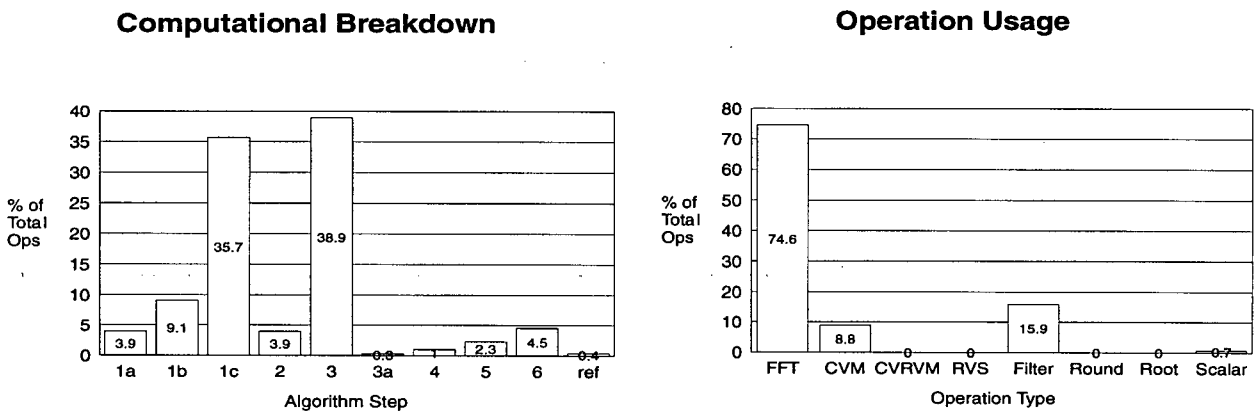


Figure 2.7: Computational analysis of SPECAN algorithm

Range-Doppler Algorithm

The number of operations required by each step of the standard range-Doppler algorithm is shown in Table 2.7. The azimuth processing steps are affected by the azimuth processing efficiency g_a.

Table 2.7: Computational analysis of range-Doppler algorithm

No.	Step in Range-Doppler Algorithm	Number of Operations
1	Range FFT	$N_{az} \cdot FFT1d(N_r)$
2	RC with SRC (multiply)	$N_{az} \cdot CVM(N_r)$
3	Range IFFT	$N_{az} \cdot FFT1d(N_r)$
4	Azimuth FFT	$\frac{1}{g_a} \cdot N_r \cdot FFT1d(N_{az})$
5	RCMC (interpolation)	$\frac{1}{g_a} \cdot N_r \cdot N_{az} \cdot FilterCR(N_{filter})$
6	AC (multiply)	$\frac{1}{g_a} \cdot N_r \cdot CVM(N_{az})$
7	Azimuth IFFT	$\frac{1}{g_a} \cdot N_r \cdot FFT1d(N_{az})$
ref	Azimuth reference function generation	$\frac{1}{g_a} \cdot \frac{N_r}{n_u} \cdot (OP_{ref} \cdot L_{az} + FFT1d(N_{az}))$

The relative number of operations required by each step in the algorithm and by each operation type is shown in Figure 2.8.

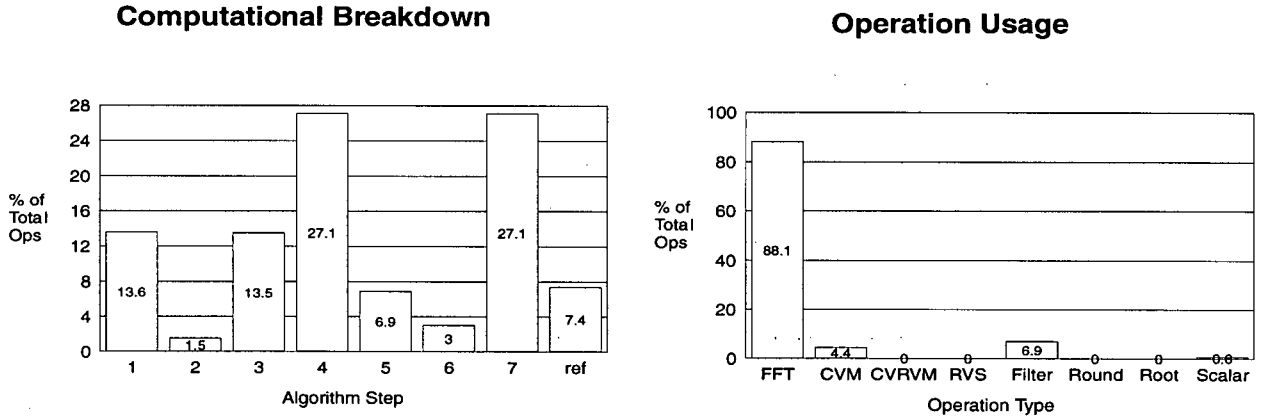


Figure 2.8: Computational analysis of range-Doppler algorithm

Wave Equation Algorithm

Table 2.8 shows the number of operations required by each step of the wave equation algorithm.

Table 2.8: Computational analysis of wave equation algorithm

No.	Step in Wave Equation Algorithm	Number of Operations
1	2D FFT	$N_{az} \cdot FFT1d(N_r) + \frac{1}{g_a} \cdot N_r \cdot FFT1d(N_{az})$
2	2D RFM	$\frac{1}{g_a} \cdot N_{az} \cdot CVM(N_r)$
3	Stolt Mapping (interpolation)	$\frac{1}{g_a} \cdot N_r \cdot N_{az} \cdot FilterCR(N_{filter})$
4	2D IFFT	$N_{az} \cdot FFT1d(N_r) + \frac{1}{g_a} \cdot N_r \cdot FFT1d(N_{az})$
ref	Reference Function Generation	$\frac{1}{g_a} \cdot \left[\frac{N_r}{n_u} \cdot OP_{ref} \cdot L_{az} + FFT1d(N_{az}) + N_{az} FFT1d(N_r) \right]$

The relative number of operations required by each step in the algorithm and by each operation type is shown in Figure 2.9.

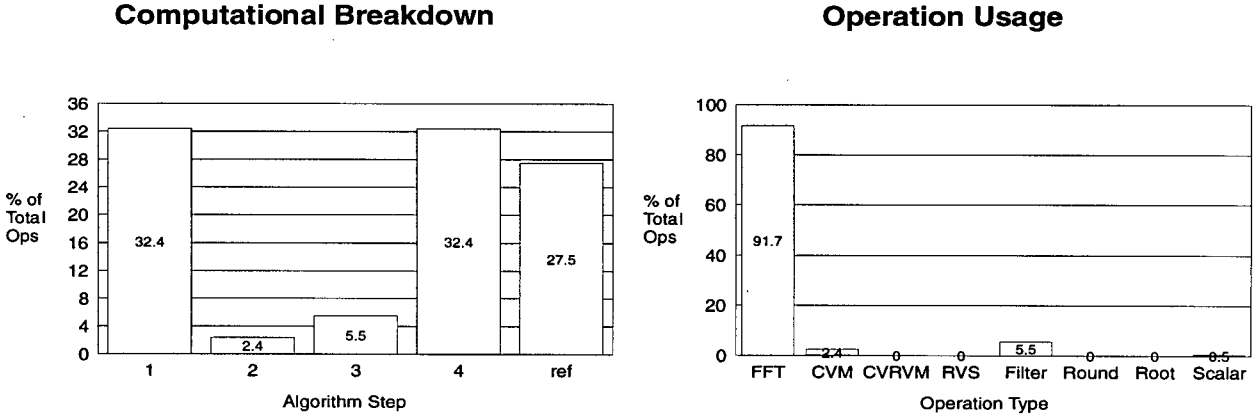


Figure 2.9: Computational analysis of wave equation algorithm

Chirp Scaling Algorithm

The number of operations required by each step in the chirp scaling algorithm is shown in Table 2.9.

Table 2.9: Computational analysis of chirp scaling algorithm

No.	Step in Chirp Scaling Algorithm	Number of Operations
1	Azimuth FFT	$N_r \cdot FFT1d(N_{az})$
2	Range Peturbation (multiply)	$N_{az} \cdot CVM(N_r)$
3	Range FFT	$N_{az} \cdot FFT1d(N_r)$
4	RC, SRC, and RCMC (multiply)	$N_{az} \cdot CVM(N_r)$
5	Range IFFT	$N_{az} \cdot FFT1d(N_r)$
6	AC and Phase Compensation (multiply)	$N_r \cdot CVM(N_{az})$
7	Azimuth IFFT	$N_r \cdot FFT1d(N_{az})$
ref	Azimuth Reference Function Generation	$\frac{N_r}{n_u} \cdot (OP_{ref} \cdot L_{az} + FFT1d(N_{az}))$
Note: All expressions are multiplied by $\frac{1}{g_a}$ to obtain the actual number of operations		

The relative number of operations required by each step in chirp scaling and by each operation type is shown in Figure 2.10.

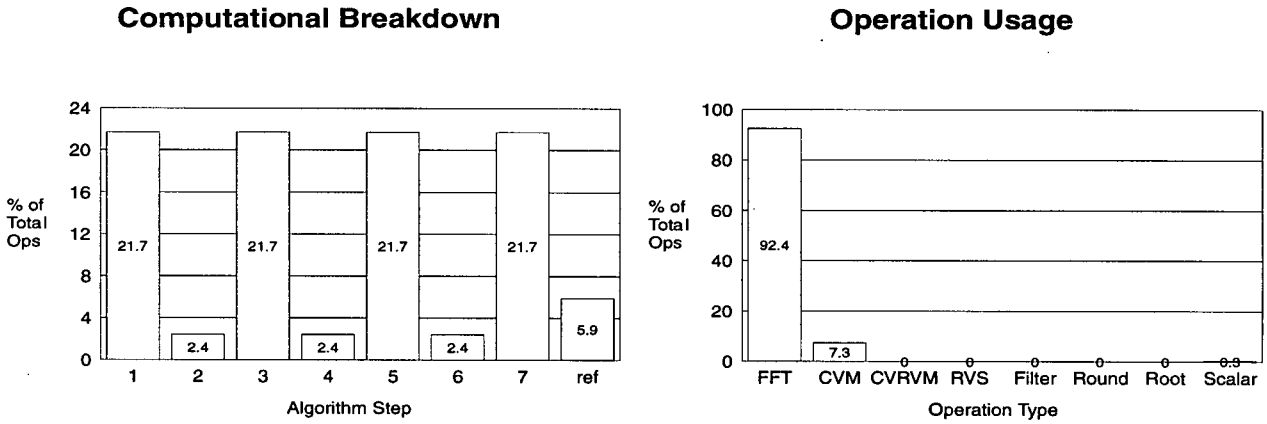


Figure 2.10: Computational analysis of chirp scaling algorithm

The wave equation and chirp scaling algorithms are made to look more computationally expensive than other algorithms in this model since the efficiency factor is applied to both range and azimuth processing while it is only applied to azimuth processing in the other algorithms. In practice, the other algorithms will also use multiple range blocks and will, therefore, suffer an efficiency penalty in the range direction.

Comparison of Algorithms

The total computation rates for each of the algorithms studied in this section are collected and shown as a function of the problem size in Figure 2.11. The vertical axis shows computation rate in GOP/s. This is calculated assuming a PRF of 1500 Hz. The horizontal axis shows the problem size in terms of n where $L_{az} = 2^n$. A somewhat different perspective is obtained by plotting the vertical axis on a log scale as shown in Figure 2.12.

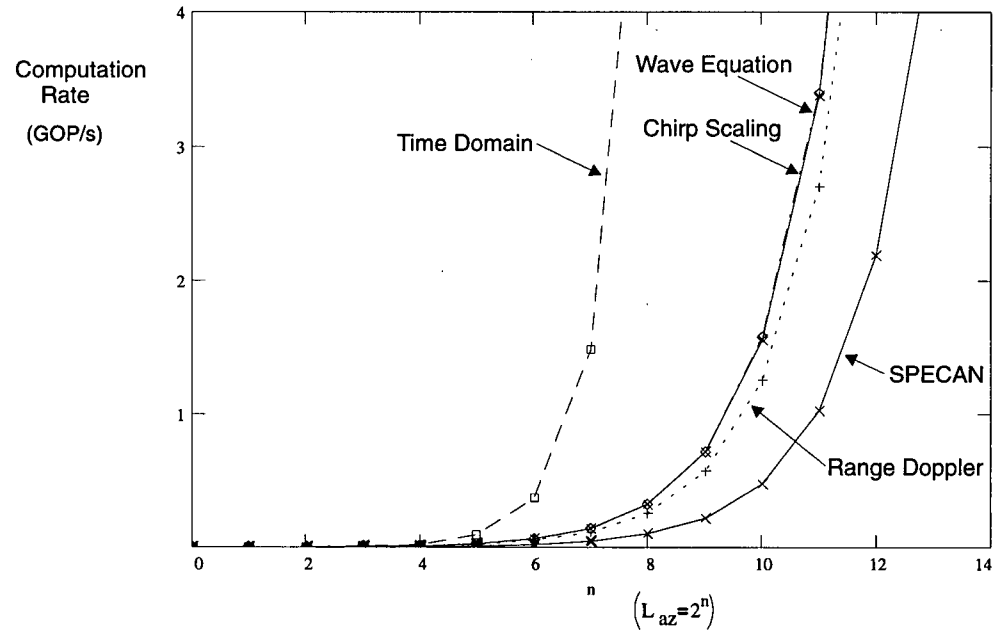


Figure 2.11: Linear scale plot of computation rate as a function of L_{az}

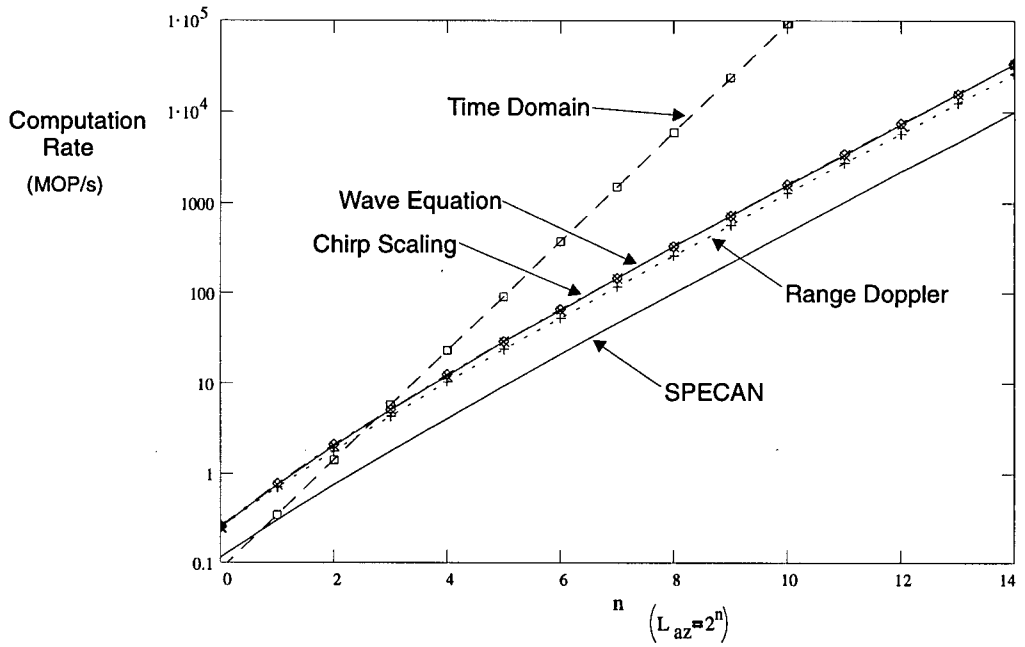


Figure 2.12: Log scale plot of computation rate as a function of L_{az}

The figures show that the time domain algorithm has requirements comparable to the other algorithms only for very short filter lengths. Once L_{az} increases past about 32, the time domain approach becomes exponentially more expensive than the other algorithms. The range-Doppler, wave equation and chirp scaling algorithms have computation rates that are similar for all values of L_{az}. As expected, SPECAN is the most efficient of all the algorithms.

The azimuth reference function length is about 2^{12} for Seasat and about 2^{10} for ERS-1 and Radarsat. This information allows approximate computation rates to be predicted for real-time processing of data from these satellites (see Table 2.10). Since the algorithm models are too simplistic to produce accurate computation rates, only relative rates are given in the table.

Table 2.10: Approximate computational requirements for real-time processing

Algorithm	Seasat (computation rate relative to SPECAN)	ERS-1/Radarsat (computation rate relative to SPECAN)
SPECAN	1	1
Wave equation, range-Doppler, chirp scaling	3.5	3
Time domain	500	200

FFTs form a major portion of all algorithms with the exception of the time domain algorithm. Figure 2.13 is a plot of the fraction of the total number of operations required by each algorithm that is due to FFTs as a function of the problem size. Figure 2.14 shows a similar plot for operations due to interpolations.

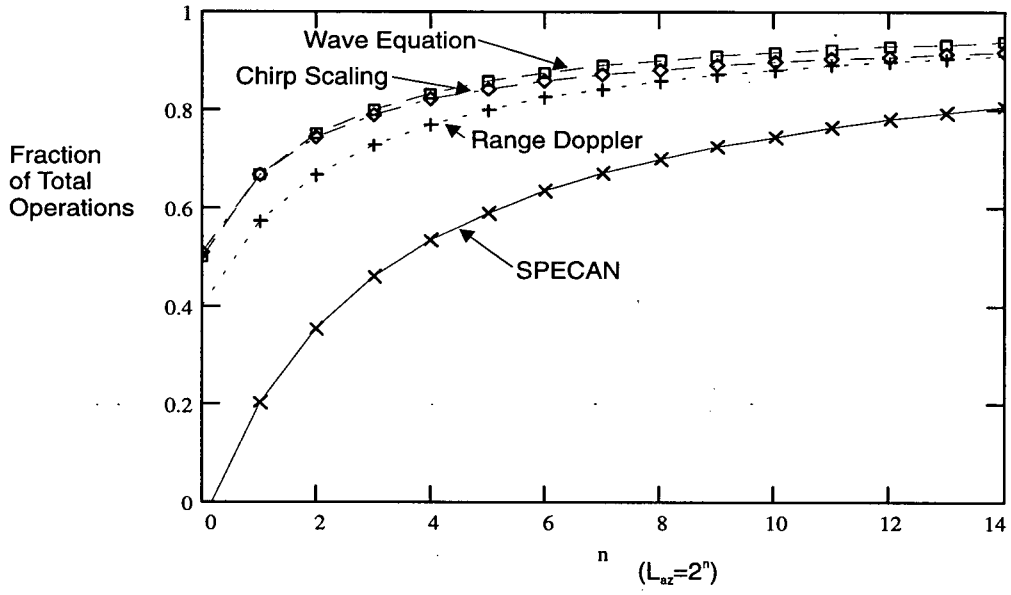


Figure 2.13: Plot of fraction of operations that are due to FFTs as a function of L_{az}

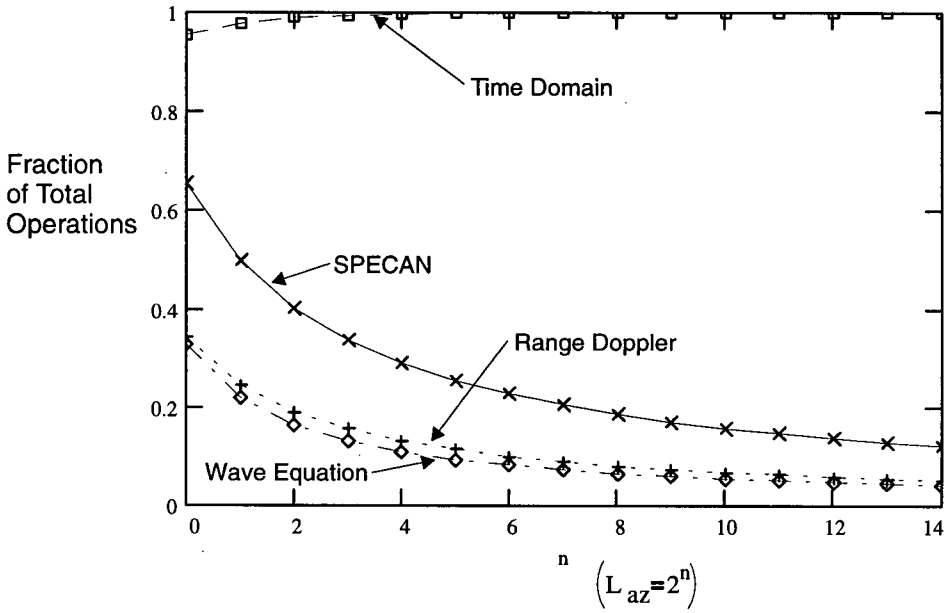


Figure 2.14: Plot of fraction of operations that are due to interpolations as a function of L_{az}

2.5.3 Model of Post-Processing Operations

Post-processing typically consists of multiplications and additions for radiometric corrections, resampling for geometric corrections, and squares and square roots for detection. These operations are common to all algorithms except the time domain algorithm. Resampling is not required in post-processing for the time domain algorithm since the resampling can be incorporated into the main processing operations. This section examines the computational requirements of the post-processing steps shown in Figure 2.5.

Table 2.11 gives the number of operations required for the post-processing steps. A resampling ratio of 1.5 is assumed in both directions. This results in output arrays that are 1.5 times as large as the inputs to the post-processing functions.

Table 2.11: Computational analysis of typical post-processing operations

No.	Step in Post-Processing	Number of Operations
1	Radiometric Correction	$N_{az} \cdot CVRVM(N_r)$
2	Azimuth Resampling	$N_r \cdot FinalN_{az} \cdot FilterCR(N_{filter})$
3	Range Resampling	$FinalN_r \cdot FinalN_{az} \cdot FilterCR(N_{filter})$
4	Detection	$FinalN_r \cdot FinalN_{az} \cdot 3$
Note:	$FinalN_{az} = Int(1.5 \cdot N_{az})$ and $FinalN_r = Int(1.5 \cdot N_r)$	

Figure 2.15 shows plots of the relative number of operations required by each step in the algorithm and by each operation type.

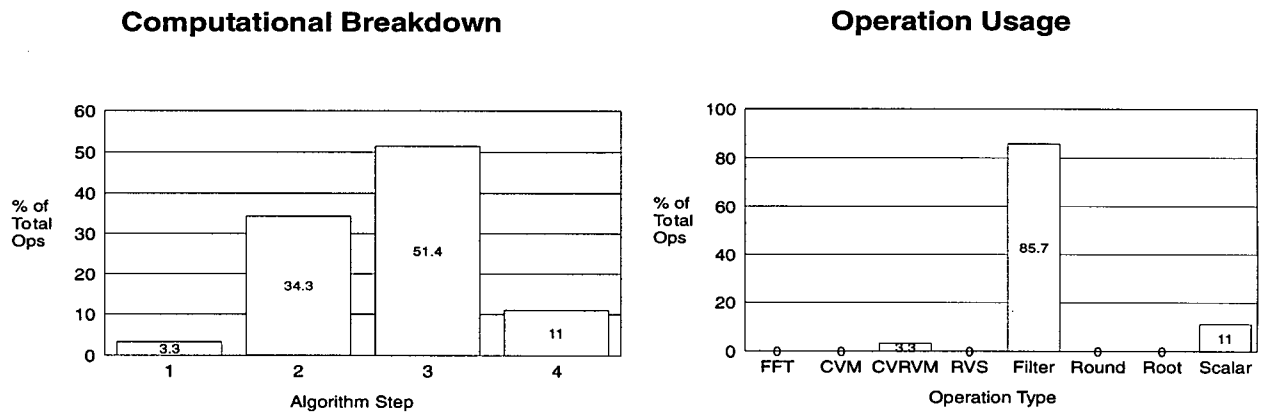


Figure 2.15: Computational analysis of post-processing operations

Figure 2.16 shows how the amount of computation in the post-processing steps grows as the data set size is increased. It can be seen that the amount of computation grows very quickly with increasing array sizes. Post-processing can form a significant portion of the total load on a SAR processor.

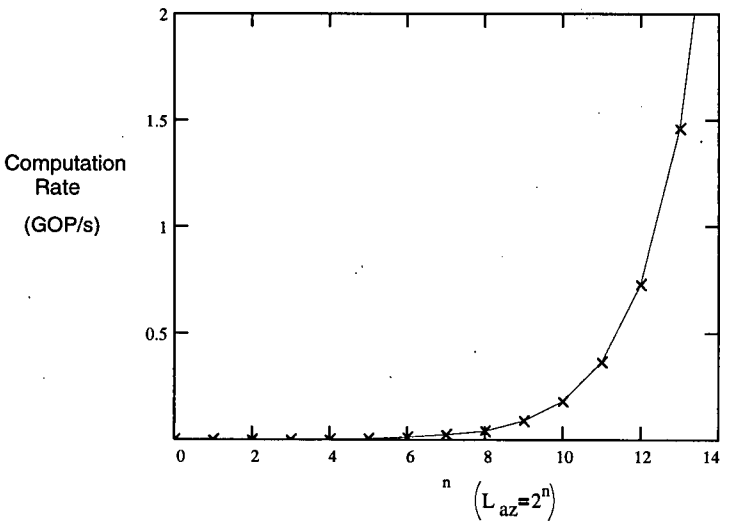


Figure 2.16: Post-processing computation requirements for varying problem sizes

The examination of post-processing in this section has only looked at a very simple scenario. More complex situations involving geocoding to a map projection and geocoding to a digital elevation model (DEM) are given in Section 9.4 of [27]. However, even in the more complex cases, the basic operations remain the same. More sophisticated post-processors make more passes over the data and use more complex address-

ing schemes for the resampling operations. Other types of post-processing operations like image compression are not considered here.

2.5.4 Model of Range-Doppler Algorithm

The previous section gave very high level models of five representative SAR processing algorithms. This section focuses on one algorithm, range-Doppler, and performs a more careful analysis of the computational and memory requirements. A more detailed model is necessary for the development of processor architectures in a later section. The range-Doppler algorithm was chosen since it is the most widely used algorithm for space-borne SAR processing. The previous section showed that the range-Doppler algorithm has computational requirements that are similar to the other high accuracy algorithms. In addition, range-Doppler is representative since it requires all of the operation types that appear in the other algorithms.

The model in this section adds more careful bookkeeping of array sizes and some post-processing steps to the previous analysis. A more detailed model of azimuth matched filter generation based on that given in [64] is used. The steps in the detailed range-Doppler algorithm model are shown in Figure 2.17. The values of the processing parameters are given in Table 2.12 and are based on typical Radarsat parameters.

Table 2.12: Range-Doppler model parameters

Parameter	Symbol	Value
Pulse Repetition Frequency	PRF	1345 Hz
Range swath length	N_r	6000
Range matched filter length	L_r	600
Range compression FFT length	N_{rfft}	8192
Azimuth matched filter length	L_{az}	600
Azimuth Compression FFT length	N_{az}	4096
RCMC interpolator filter length	N_{rcmc}	8
Resampling filter length	N_{res}	8
Oversampling ratio for resampling	$OVSM$	1.4
Azimuth filter update interval	n_u	1
Computation word length	N_{bits}	32

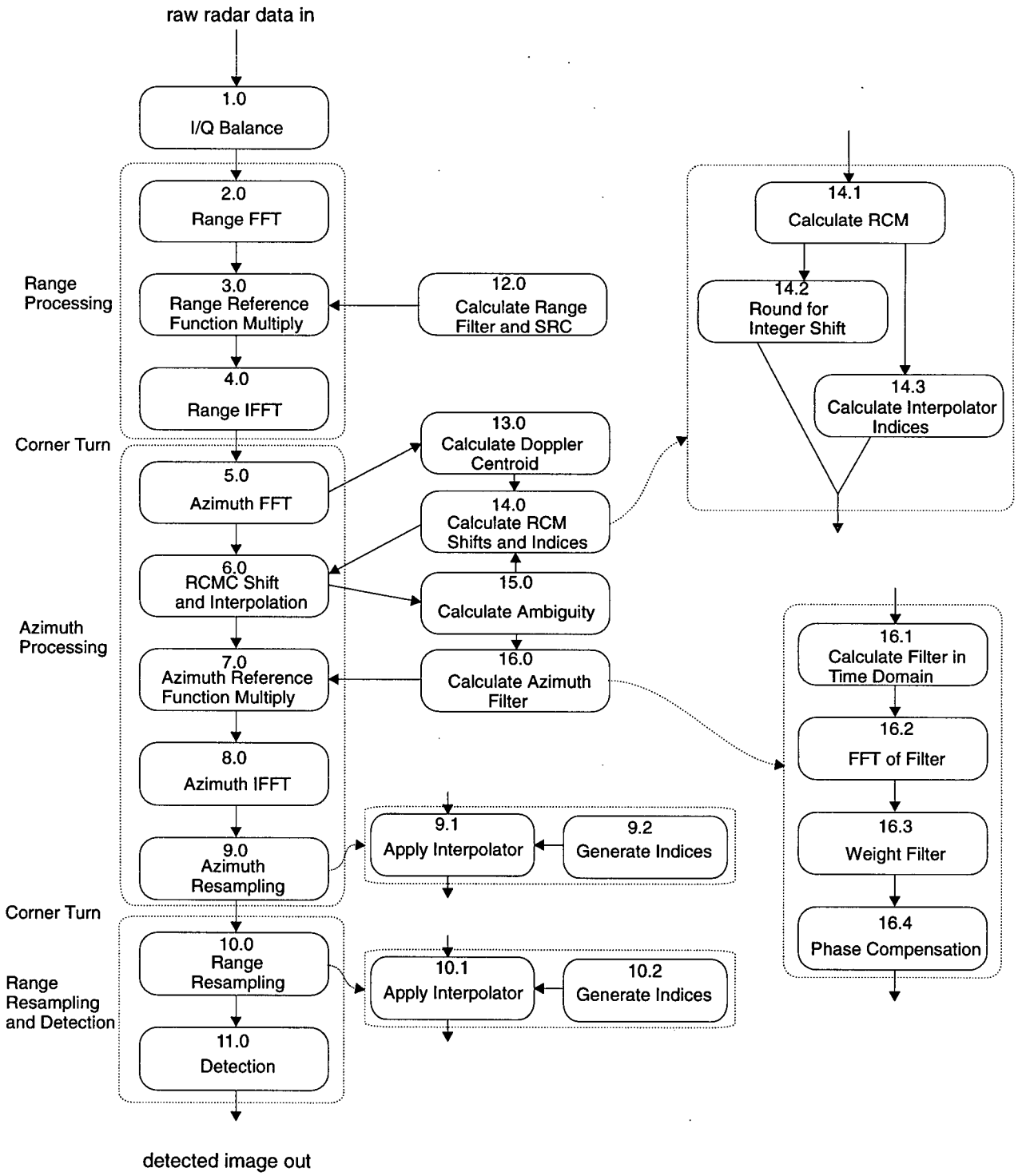


Figure 2.17: Flow diagram of range-Doppler algorithm

The range-Doppler algorithm was modelled using a Mathcad document. The key portions are shown in Appendix A. The amounts of computation and memory required by each bubble in the flow diagram of Figure 2.17 was calculated. The number of operations required by each of the major steps in the algorithm is shown in the first graph in Figure 2.18. The adjoining graph shows the portion of the total computation that is due to each of the basic operation types.

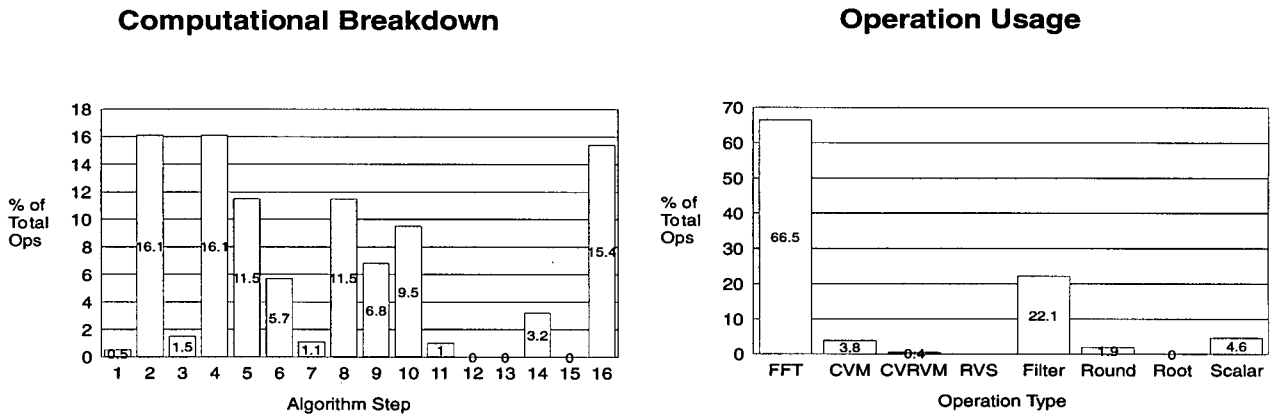


Figure 2.18: Computational analysis of detailed range-Doppler algorithm

Figure 2.18 shows that the more detailed model has a similar computational breakdown as the simpler model whose results are shown in Figure 2.8. One of the main differences is the additional processing that is required by the azimuth and range resampling operations which account for about 17% of the total. The model presented here is in some ways a worst case model since n_u was set to one. This means that a new azimuth reference function is calculated for each azimuth line. The memory requirements for the algorithm vary during the course of execution from a minimum of 140 MB at the output to 280 MB before detection. It is possible to reduce the maximum value by using shorter FFT lengths, by combining some algorithm steps, or by using a shorter wordlength.

2.6 Partitioning of SAR Processing

The high computational requirements for SAR processors result in a need to partition the SAR algorithm to run on multiple processors. This section examines the partitioning of SAR algorithms and provides a more general treatment of this topic than is available in the literature. Section 2.6.1 discusses some characteristics of SAR processors that affect partitioning. Section 2.6.2 looks at the issue of granularity. Section 2.6.3 presents a classification of coarse grain partitioning approaches.

2.6.1 Considerations for Parallelism

Some of the key design issues that need to be addressed when partitioning a SAR algorithm for parallel processing are listed below.

- **granularity of parallelism:** Is parallelism exploited at a fine grain level, at a coarse grain level, or both?
- **choice of horizontal and/or vertical parallelism at both the fine and coarse grain levels:** What combination of horizontal and vertical parallelism is used?
- **data partitioning scheme:** How is the data divided between the processors?
- **amount of parallelism:** How many processors are used? What is the size of the data partitions and how many of them are there?

Some of the options in these areas are discussed from an algorithmic perspective in the sections that follow. Specific examples of these characteristics are given when actual processor architectures are studied in Section 9.

Some of the characteristics of an algorithm that affect the partitioning in a parallel SAR processor are listed below.

- **structure of the processing flow diagram:** This determines the number and type of operations, data dependencies, as well as the scheduling and synchronization requirements. Most SAR algorithms decompose naturally into large sequential steps with fairly simple data dependencies and synchronization requirements.
- **data array sizes in range and azimuth directions:** Large arrays have higher computational requirements but allow for more parallelism.
- **reference function lengths:** The long reference function lengths required for high accuracy processing and the resulting overlaps required between data partitions have an influence on efficiency.
- **reference function update intervals:** The filter update intervals can place upper bounds on data partition sizes. In SAR, this applies especially to the partitioning of azimuth compression since reference function updates are much more frequent during azimuth processing than during range processing.

2.6.2 Granularity of Parallelism

The granularity of a parallel system is the size of the units by which work is allocated to processors. For SAR processing, two general levels of granularity can be established. The bulk of SAR processing is made up of DSP operations like FFTs and vector multiplies. *Fine grain parallelism* can be considered to be parallelism that subdivides these fundamental operations. *Coarse grain parallelism* allocates each fundamental operation to a single processor.

Coarse grain parallelism in SAR processing offers the advantages of simpler scheduling, fewer synchronization problems, and the ability to use commercial system components. In contrast, fine grain parallelism offers the potential of more parallelism and, therefore, higher speed-ups, and reduces load balancing problems. However, fine grain parallelism requires low communication times between processing elements (PEs), and the lack of suitable commercial components usually implies custom VLSI designs.

Both approaches have potential application in SAR processing. Some of the characteristics of SAR processing that make it suitable for coarse grain parallelism are as follows:

- SAR requires large data sets that can easily be partitioned among PEs.
- SAR algorithms are composed of distinct sequential large grain blocks that can be separated and pipelined.
- Computationally expensive algorithm steps consist of standard DSP operations for which standard components exist.

SAR is also suited for fine grain parallelism:

- Most algorithm steps consist of standard DSP operations for which parallel structures are well known.
- The primary data flow in SAR processing has no low level recursive structures like IIR filters. The loops that do occur are at the highest level of the algorithm and involve less computationally expensive operations. This facilitates fine grained parallelism since it results in fairly simple data dependencies. As a result, scheduling and synchronization are greatly simplified.
- The vector lengths are generally quite large so very long fine grained pipelining can be used with little overhead.

Fine grain parallelism is potentially a very attractive approach with the promise of large speed-ups. As a result, the area has attracted a great deal of research. Some areas, like parallel FFTs, have been studied in great depth (see for example [43]). However, due to the high cost of implementing an algorithm in a fine

grained parallel fashion, this approach is more suited to custom systems with special requirements. The high cost is mainly due to the lack of existing hardware and software components for implementing such systems. Most of the developments in the computer industry have focused on developing systems based on microprocessors which are coarse grain components. The lack of mass-produced fine grained parallel processors, industry standards, and software infrastructure have led to fine grained systems being limited to custom designs or research projects. Since one of the attributes of a good SAR processor architecture is low development cost, much of this thesis will focus on coarse grain approaches. However, some consideration will also be given to certain fine grained architectures.

2.6.3 Classification of Approaches to Partitioning

This section gives a classification of common coarse grain partitioning approaches for SAR. The classification is based on the scheme given in Section 5.2 of [15]. There are many characteristics that need to be considered when classifying parallelizations of algorithms, and no one set covers all situations. The partitioning approaches in this section should be taken as starting points for describing a specific design. The approaches considered are:

- vertical partitioning,
- horizontal and horizontal-vertical partitioning, and
- vertical-horizontal partitioning.

These approaches are described in the paragraphs that follow. The partitioning approaches are also illustrated in Figures 2.20 through 2.22. The meanings of the symbols that are used in these figures are shown in Figure 2.19.

Vertical Partitioning

The first approach to partitioning SAR processing is vertical partitioning, which is also known as pipelining or temporal parallelism. Figure 2.20 shows an example of vertical partitioning. The granularity of the partitioning can be varied by changing the number of pipeline stages. This form of partitioning increases the throughput of the processor but increases its latency. Pipelining is well suited to SAR processing since most SAR algorithms consist of a number of sequential steps.

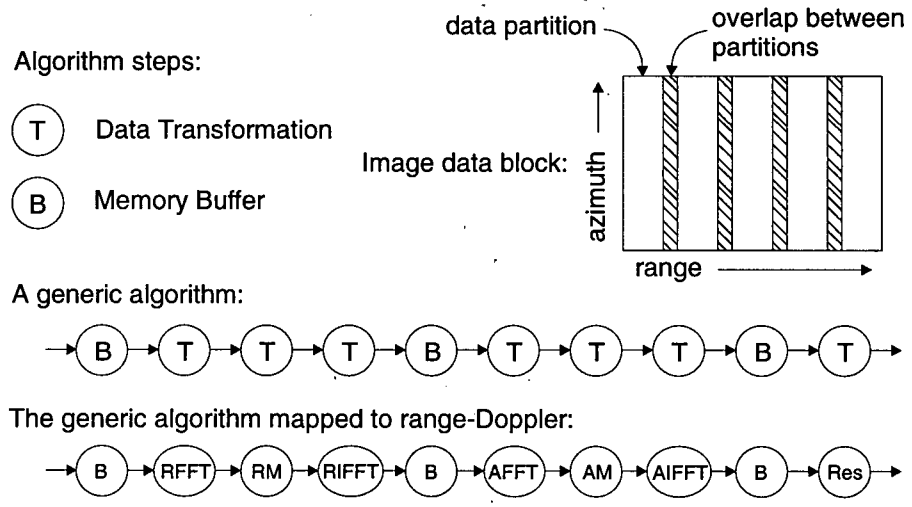


Figure 2.19: Symbols used to depict approaches to parallelism

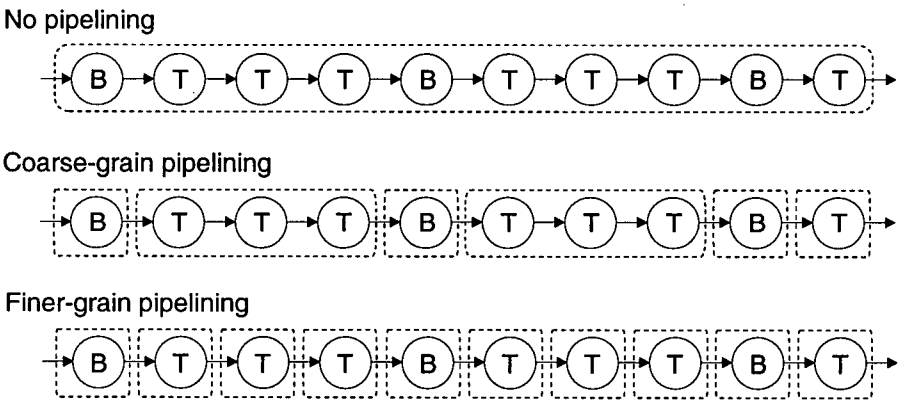


Figure 2.20: Vertical partitioning

Horizontal Partitioning

The next approach is horizontal partitioning or data parallelism as shown in Figure 2.21. Horizontal partitioning divides the data set among the processors. Each processor performs essentially identical operations on subsets of the data. There are a number of options that can be applied to horizontal partitioning:

- the use and granularity of pipelining in each of the parallel processors:
  - none (strictly horizontal partitioning)
  - pipelining with some amount of granularity (horizontal-vertical partitioning)
- the orientation of the data partitioning:
  - azimuth subswaths or strips
  - range subswaths or strips

- submatrices

- the size of the data partitions: Smaller partitions will allow for more parallelism but will result in decreased efficiency.

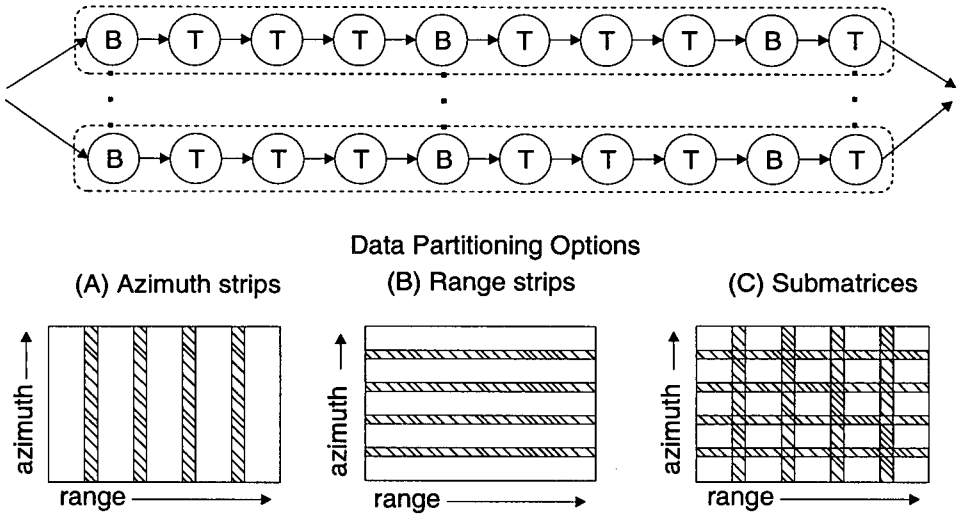


Figure 2.21: Horizontal partitioning

The data partitioning can be static or dynamic. In the static case, each PE operates on the same data set throughout. In the dynamic case, some or all of the data is exchanged between PEs during processing. Static partitioning makes use of distributed corner turns. Each processor has enough data to perform processing in both range and azimuth directions. This places a lower limit on the size of the data partitions since the partitions must be at least as long as the reference functions in each direction. In a dynamic partitioning, the option exists to perform corner turn operations in which PEs exchange data until each has a data set suitable for processing in the orthogonal direction.

Vertical-Horizontal Partitioning

Vertical-horizontal partitioning is a combination of the previous two partitioning methods. The processing is partitioned vertically through the use of pipelining and horizontally by dividing the data among the processors. This approach can be used to take advantage of the fact that in many SAR algorithms, range processing is performed first, then the data is corner turned, and then azimuth processing is performed. The corner turn buffer is a natural synchronization point for the parallel processors. Vertical-horizontal partitioning and some of its data partitioning options are shown in Figure 2.22. Some of the options that can be chosen are as follows:

- the use and granularity of pipelining in the individual processing sections

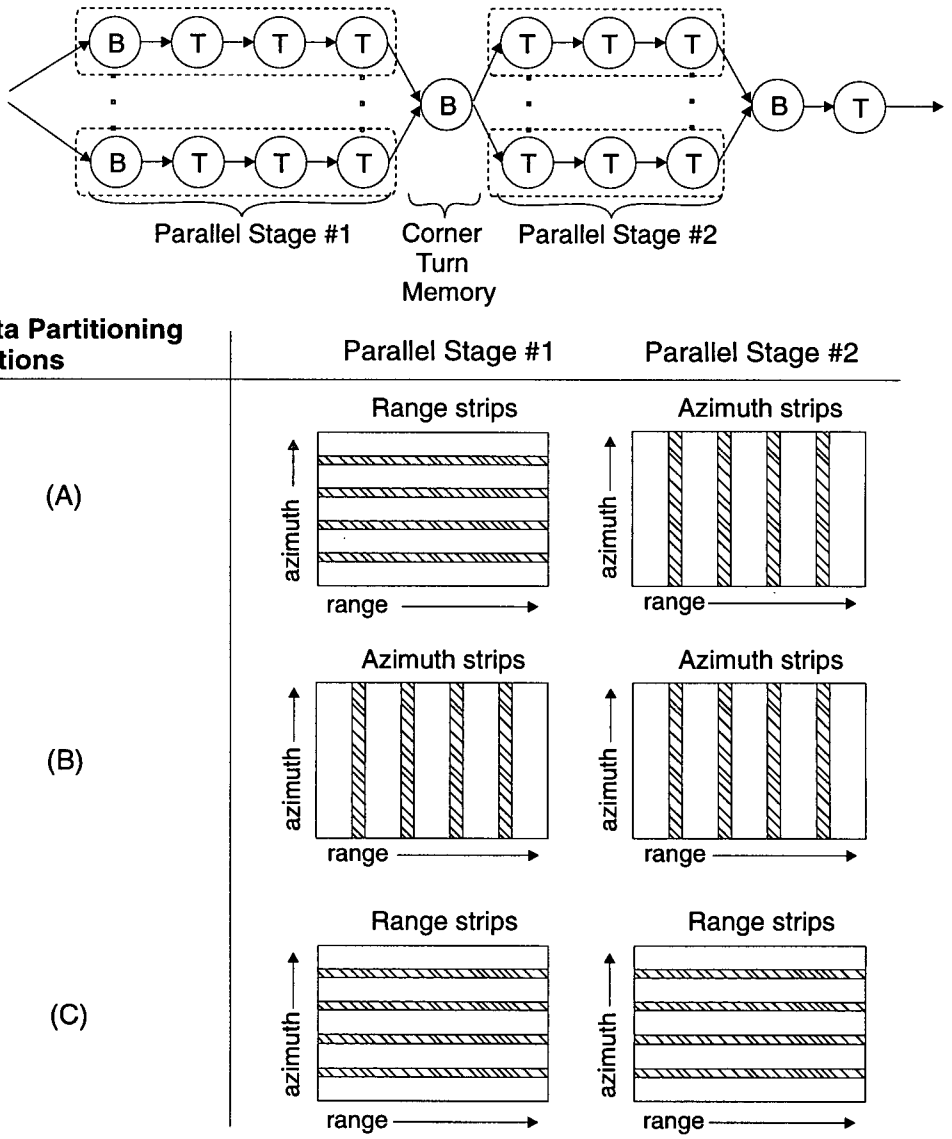


Figure 2.22: Vertical-horizontal partitioning

- data partitioning:
  - range-azimuth
  - azimuth-azimuth
  - range-range
  - submatrices or other options
- the size of the data partitions

2.6.4 An Example of Horizontal Data Partitioning: Range-Doppler Algorithm

The range-Doppler algorithm described in Section 2.5.4 is partitioned into range subswaths and analyzed in this section. Each range line is split into sublines with enough overlap to account for the matched filter length and RCMC overlap. The range FFT size is chosen to be the next largest power of two. The range subswath widths and range FFT sizes are shown in Figure 2.23. The operation rate required for each subswath processor is plotted in Figure 2.24. The graph also shows the total amount of computation which is the sum of the computation done by each processor. The adjoining graph plots the efficiency as a function of the number of processors. It can be seen from the graphs that increasing the number of processors within regions where the range FFT length is constant decreases the efficiency significantly because the workload per processor does not decrease very much. The overhead imposed by the overlapped regions imposes a limit on the parallelization of the algorithm (see equation 2.24 for the range processing efficiency g_r).

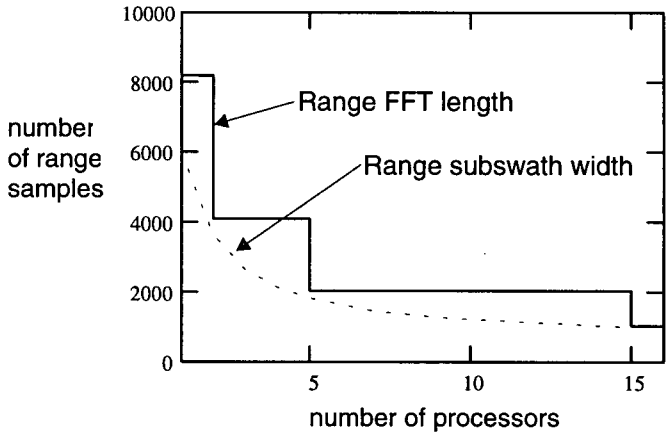


Figure 2.23: Range block size

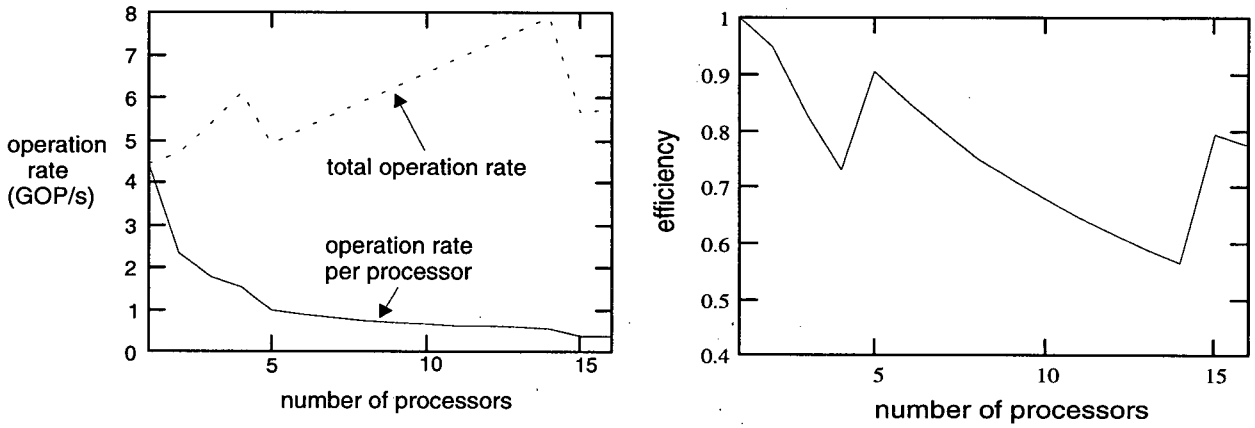


Figure 2.24: Parallel operation rates and efficiency

The memory requirements of a processing system also increase as the number of processors is increased. A plot of the required size of the input buffer memory is plotted against the number of processors in Figure 2.25. It can be seen that the total amount of memory required rises linearly with the number of processors.

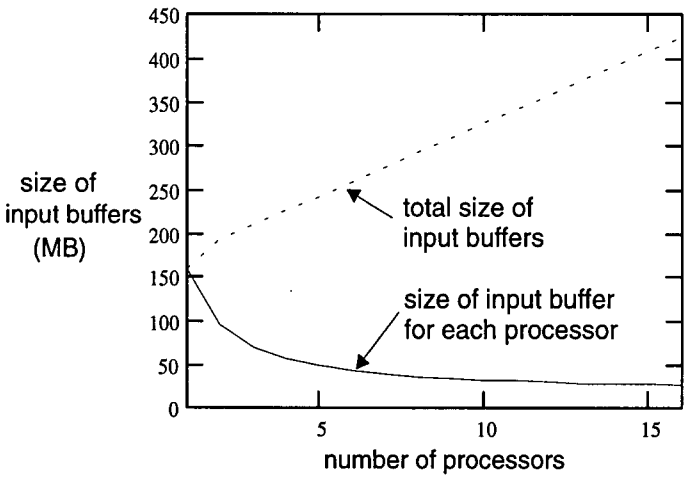


Figure 2.25: Input buffer memory size

The restraints on parallelization imposed by the matched filter lengths in a static horizontal partitioning lead us to look for ways around this limitation. Some possible ideas and their drawbacks are:

- use dynamic data partitioning to add a global corner turn step: The amount of inter-processor communication will increase.
- use shorter filter lengths: The accuracy of the processor will suffer.
- use vertical-horizontal partitioning: The amount of inter-processor communication will increase and a centralized corner turn memory may become a bottleneck.
- use an algorithm like SPECAN that does not require overlaps: SPECAN has precision limitations and has a more complicated data partitioning scheme.
- use fine grain parallelism: The system design may be more difficult.

3 SAR Processor System Design

The SAR processing problem was outlined in the previous section. This section begins the process of designing a processor that provides a solution for that problem. SAR processors are large complex systems and, therefore, there are many issues that need to be considered during the design process. This section looks at some of these general design issues in preparation for a more specific look at SAR processor designs in the chapters that follow. General overviews are given of system design methodology (Section 3.1), system requirements (Section 3.2), selected design considerations (Section 3.3), and performance prediction (Section 3.4). Finally, the architecture selection criteria used in this thesis is presented (Section 3.5). The intent is not to cover these topics in depth but rather to identify them as important considerations in SAR processor design.

A SAR processor is part of a very large system that in its entirety includes everything from the SAR sensor to the display, storage, and dissemination systems for the fully processed products. This thesis only looks at the signal processing subsystem. None of the other system components are considered. These components can include the communications system, frame synchronizer, tape drives, disks, display systems, backup and archival systems, higher level control functions, user interfaces, and other external interfaces.

3.1 Design Methodology

Since SAR systems tend to be large and complex, methodologies used for designing other complex systems are applicable. Methodology is a large topic and the discussion here is just a brief look at the steps involved in designing a SAR processor. More detailed information on signal processor design methodologies can be found in [13][15]. The general process of designing a SAR processor can be summarized as follows:

- define the processor requirements,
- select algorithm and evaluate image quality and computational loading,
- survey candidate signal processor architectures and the available technology,
- design and evaluate several candidate logical architectures, and
- select an architecture and perform detailed design.

Throughout the design process, a constant evaluation of design trade-offs needs to take place. The evaluation is based on cost/performance trade-offs that consider both implementation and maintenance costs.

The ever-shrinking design times have created a need for improvements to the design process. The concepts of rapid prototyping and hardware/software codesign attempt to provide solutions to this problem [62]. For concepts like this to be useful, they need to be supported by good tools. Many tools exist that aid in the design of algorithms (e.g. Matlab). Some tools exist that support limited versions of codesign (e.g. Ptolemy [56]). Both software design and detailed hardware design are well supported by existing tools. Fewer tools exist to aid in the development of architectures.

An ongoing program exists in the United States that will have an impact on how embedded SAR processing systems are designed in the future. The US Department of Defence Advanced Research Projects Agency (ARPA) program for Rapid Prototyping of Application Specific Signal Processors (RASSP) aims to significantly improve the process by which embedded digital signal processors are developed [81]. Some of the main directions being taken by the RASSP program are given below.

- executable requirements [3]
- virtual prototyping: software models of hardware
- co-development
- standard hardware interfaces
- reuse libraries

The RASSP program is specifically aimed at improving the design process of systems like high-performance SAR processors. The directions taken by this program are likely to be representative of methods that will be used in future SAR system designs.

3.2 System Requirements

As for any system, a SAR processing system needs a complete set of requirements. Some basic algorithm related requirements for a SAR processor and their effects on system design are as follows:

- **radiometric accuracy:** affects algorithm, number representation, post-processing corrections, etc.
- **resolution requirement:** affects algorithm, the amount of data, computation, memory, etc.
- **choice of SAR algorithm:** choice of processing parameters, data set sizes, filter lengths, number of looks, etc.

These areas are covered in Section 9 of [27].

The system requirements also need to address overall system functionality, external interfaces, performance, cost, and other operational factors like reliability, maintainability, manability, supportability, and economic feasibility [13]. The goal in defining the system requirements is to ensure that the final system meets the needs of the customer.

3.3 System Design Considerations

This section summarizes some of the general design considerations that must be addressed by every SAR processor design irrespective of the requirements of the specific system. All of the issues discussed here present the designer with decisions and trade-offs. Many of the decisions are interrelated. There is no cookbook recipe for coming up with an optimal design. Numerous high-level design iterations are necessary before an optimal solution can be found. A reference on systems engineering should be consulted for a broader look at all the issues of designing a complex system [13].

One of the most fundamental trade-offs in signal processor design is the one between performance and generality/flexibility. The more general the system the lower its performance. It is always possible to achieve higher performance if the architecture is tuned to one algorithm and associated set of processing parameters. Performance decreases if the architecture is made more flexibility to algorithm and processing parameter changes. This is not an easy trade-off to make since it involves many implementation and lifecycle issues.

Some implementation issues in each of the three areas, algorithm, architecture, and technology, are listed below:

Algorithm

- **fast algorithms:** A multitude of options exist for choosing fast ways of implementing the basic DSP operations. For example, there are at least a dozen ways of implementing FFTs [12]. Each of these is suited for different situations and makes a unique set of trade-offs. For each of DSP operations in the SAR algorithm, an algorithm needs to be chosen or a new one developed. Different number systems or other number theoretic techniques may be used to speed up the computations.
- **arithmetic:** Many options also exist for the method of performing arithmetic in the system. Some options are parallel, bit-serial, distributed arithmetic or some combination of these.

- **number representation:** The way numbers are represented in the system is a major factor in system performance and memory sizes. A fixed, floating point or block floating point representation with some word length needs to be chosen. A careful analysis of the number representation's effect on accuracy, truncation errors, memory sizes, and arithmetic unit complexity must be made.

Architecture

- **data transfer rates:** Both external and internal transfer rates need to be determined and carefully allocated.
- **amount and type of parallelism:** The optimal level of parallelism must be decided upon based on a cost/performance analysis.
- **processor selection/design:** The selection or design of the processing elements will determine the basic system building blocks.
- **memory organization and capacities:** The types, sizes, and interconnection of all the memory types in the system present a wide range of design options.
- **general design principles:** Principles like modularity, flexibility, and the others mentioned in Section 3.2 must be considered during the design of the architecture.

Technology

- **use of COTS vs. custom hardware:** In general, users prefer not to buy custom hardware if their requirements can be filled by standard computer system components.
- **implementation technology independence:** It is advantageous, if future technology enhancements can be used to enhance the system rather than make it obsolete.
- **the technology freeze date:** This sets the date by which all system components must be available; newer components are excluded from consideration.
- **tolerable level of technology risk:** It is always riskier to use new cutting edge technologies or products in a system even if they offer higher performance than older more established products.

3.4 Performance Prediction

One of the key tasks in SAR processor design is predicting the performance of candidate architectures. Unfortunately, performance prediction can be very difficult. This is especially true for general purpose computers. Usually the more general the system, the more difficult performance prediction becomes. Performance analysis is covered in detail in [52].

One of the most common methods of predicting performance is the use of standard benchmark programs. A large number of benchmarks exist. However, most are aimed at general purpose computers and not at DSPs. There are currently some efforts under way to set up a standard set of DSP benchmarks (see for example [94]) but few performance results are available. Most general benchmarks are not very accurate at predicting actual performance for DSP applications. The only truly accurate performance estimates are obtained when actual SAR code is executed on a processor.

One of the few DSP benchmarks that has gained any widespread use is the time for a 1k complex FFT. Given SAR's dependence on FFTs and the lack of other benchmarks, the 1k CFFT time will be used extensively in the remainder of this thesis. The 1k CFFT time is an important measure of the performance that can be expected from a given processor when running a SAR application since a large portion of the run time of most SAR algorithms consists of FFTs. However, it is dangerous to put too much importance on the 1k FFT times. Certain processors may be optimized for FFTs and, therefore, have excellent FFT times but show very poor performance on other operations like filtering. Also, the FFT times for a range of vector sizes should be looked at since the execution time may be strongly influenced by the data set size. This is usually due to the effects of caching.

Difficulties arise when comparing machines that have different levels of generality. Performance measures must be looked at in context of the overall system capabilities. Thus, it is not appropriate to compare a general purpose machine to a special purpose or dedicated processor on only one benchmark result.

3.5 Architecture Selection Criteria

In the sections that follow, many possible processor design approaches are presented. Some criteria are necessary to select the ones worthy of further study. This is best accomplished by establishing requirements that must be met by the SAR processor. The four primary requirements are as follows.

Performance

The processor shall be capable of processing full resolution satellite SAR data at 1/10 of the real-time rate under the processing scenario defined in Section 2.5.4. Alternatively, it shall be possible to process data at a lower resolution at the full real-time rate.

Scalability

It shall be possible to scale the processor up to handle real-time full resolution processing with an approximately linear increase in cost.

Flexibility

The processor shall be flexible enough to accommodate a number of possible SAR processing situations. Processors based on frequency domain processing shall be capable of handling any of the four frequency domain based algorithms described in Section 2.4. Processors based on either time or frequency domain approaches shall be capable of handling changes in any of the processing parameters.

Development cost

The development cost shall be minimized by making use of existing hardware and software products wherever possible.

Repeat cost

The repeat cost (the cost of additional SAR processors) shall be minimized.

One tenth real-time processing is chosen as the primary performance requirement since this level of performance is much more attainable by current technology than full real-time operation and is, therefore, much more cost effective. The lower cost of such processors also implies a larger potential market.

In some cases, minimizing the development cost and the repeat cost are conflicting requirements. The intent here is to find a reasonable balance between these costs while meeting the other requirements.

Some examples of other criteria that are not considered to be of primary importance in this work are power consumption, weight, and volume.

4 Architectural Approaches

This section describes some general categories of SAR processor architectures. There is no one classification that covers all possible architectures. However, the one given here has been adapted from existing classifications to handle the most common SAR processing situations. The following list gives the classifications as well as their relation to Flynn's taxonomy of parallel computers [37].

- single processor (SP) - SISD
- common node processor (CN) - MIMD
- multiprocessor (MP) - MIMD
- pipeline processor (PL) - MIMD or pseudo-MISD
- multicomputer (MC) - MIMD
- SIMD processor - SIMD
- Hardwired (HW)

Many SAR processors make use of a mixture of these approaches. One architecture may describe the high level organization of the system, while another the lower level components, which are themselves parallel computers. More detailed information on the large variety of parallel architectures can be found in [29]. In all cases, a wide range of performance levels can be attained by each of the architectural approaches. Performance depends largely on the number of PEs, the choice/design of the individual PEs and the amount of memory associated with each.

4.1 Single Processor

Single processor architectures offer the simplest and cheapest SAR processing solution. However, due to the large computational requirements, uniprocessors are only suited for low performance applications. As microprocessor performance increases, uniprocessors are becoming useful for more and more SAR applications. An example of such a system is the running of SAR processing software on a modern high performance workstation with a high speed RISC processor and a large main memory and disk capacity.

4.2 Common Node Processor

Like a uniprocessor, a common node processor has a single PE through which all data must pass at various points during execution. However, multiple processors are available to assist the common PE in a processor farm configuration. A typical common node architecture is shown in Figure 4.1 Common node proces-

sors differentiate themselves from multiprocessors and multicomputers in that they have a single PE that distributes and collects the SAR data from the other PEs. Common node architectures have in the past been the most common way of building medium performance SAR processors.

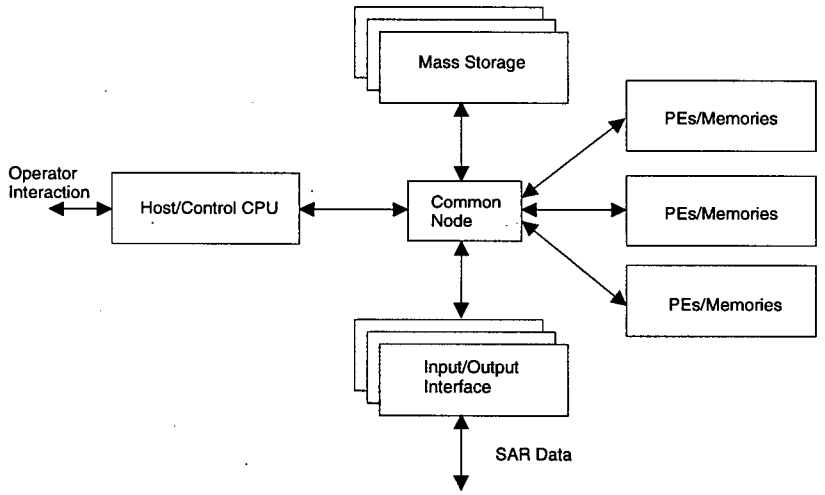


Figure 4.1: Typical common node architecture

Some advantages of common node architectures are listed below.

- Common node architectures are flexible in handling algorithmic changes as well as different algorithms.
- Most or all of the system components are usually available off-the-shelf.
- Common node systems are usually not much more difficult to program than uniprocessors.
- There is much existing experience in building such systems.

Some disadvantages of common node systems are given below.

- Scalability is a problem. The common PE can be a bottleneck that prevents the performance of the system from being increased.
- Performance prediction can be difficult.

An example of a common node processor is a minicomputer with an attached accelerator like an array processor. The accelerator may also be a parallel processor and may be designed using a different architecture (e.g. multicomputer). Another example of a common node processor is a network of workstations with a single master workstation. More examples can be found in Section 9.3 of [27].

4.3 Multiprocessor

A multiprocessor architecture consists of multiple processors that access a shared memory. Caches are typically used to reduce the number of accesses to the shared memory. A typical multiprocessor architecture is shown in Figure 4.2.

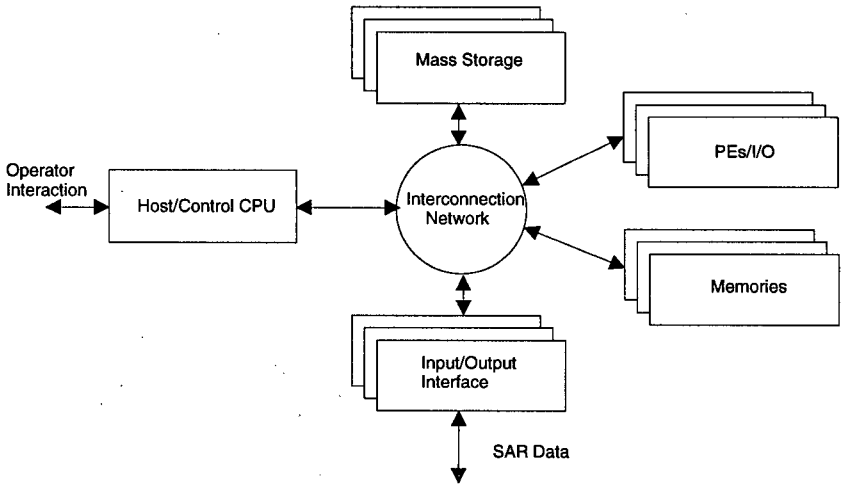


Figure 4.2: Typical multiprocessor architecture

Some advantages of the multiprocessor approach are listed below.

- Multiprocessors are flexible. They can handle algorithmic changes as well as different algorithms.
- Either a complete off-the-shelf processor or off-the-shelf PEs can be used.
- Multiprocessors use the same familiar global memory programming model as uniprocessors.

Some disadvantages of multicomputers are given below.

- Scalability is a problem. The shared memory can be a bottleneck that prevents the performance of the system from being increased.
- Performance prediction can be difficult.

Multiprocessors are likely the most common type of parallel machine being built today. Terms like tightly-coupled, shared-everything, or symmetric multiprocessing (SMP) are used to refer to multiprocessor variants. Examples include the multiprocessor workstations available from vendors like DEC, HP, IBM, SGI and SUN.

4.4 Pipeline Processor

High performance SAR processors have traditionally been constructed as pipeline processors. Figure 4.3 shows a high-level diagram of a pipeline processor.

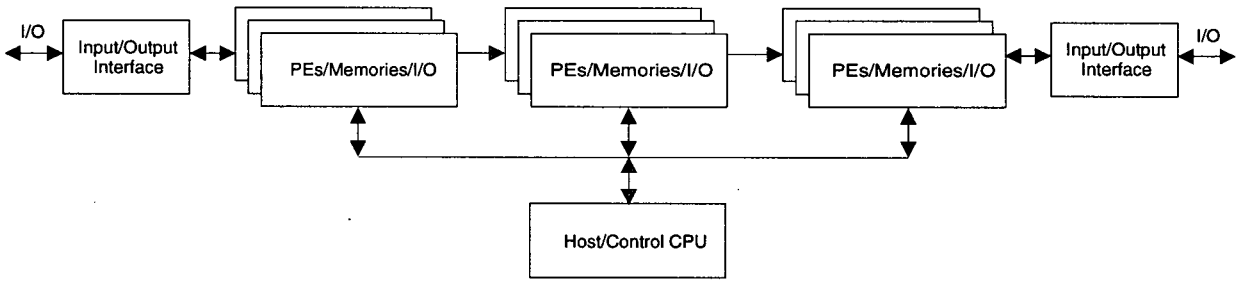


Figure 4.3: Typical pipeline architecture

Pipelining is often combined with other architecture approaches in the design of SAR processors. For example each pipeline stage can be designed using a multicomputer architecture. In general, each stage of the pipeline can itself be a parallel processor that is designed using any of the architectures in this section.

A key characteristic of pipeline processors is the amount of flexibility designed into the system. Hardwired or fixed function pipelines have PEs and interconnections that are fixed and dedicated to a single algorithm. Programmable pipelines have programmable PEs and a more general interconnection scheme that can be adapted to a number of algorithms. Hardwired pipelines tend to be faster while programmable pipelines are more general.

Some of the advantages of the pipeline approach are listed below.

- A pipeline exploits the sequential nature of the steps in the algorithm.
- Off-the-shelf processing elements (PEs) can be used.
- The performance of a pipeline processor is relatively easy to predict.
- There is a great deal of prior experience in building pipeline SAR processors.

The following is a list of some of the potential problems with pipeline SAR processors.

- The processor as a whole will most likely not be available off-the- shelf.
- A specialized pipeline may not be flexible enough to handle changes in the algorithm or other aspects of SAR processing.
- Buffers are needed between all the stages.

- Expandability can be expensive since all the stages need to be expanded together.
- Fault tolerance may be difficult to achieve. In a simple pipeline, each PE or buffer is a critical link.
- Load and I/O rate balancing can be difficult.

The NASA JPL Advanced Digital SAR Processor (ADSP) and its derivatives are hardwired pipelines. These processors exhibit extremely high performance but only for a specific algorithm. VASP based SAR processors like the MDA IRIS X2C can be described as programmable pipelines in which each pipeline stage is a multicomputer composed of Motorola 96000 DSPs.

4.5 Multicomputer

The most flexible type of parallel architecture is the multicomputer, which is composed of a distributed collection of PEs with local memories. Multicomputers are gaining in popularity due to their flexibility and ability to use off-the-shelf components. Figure 4.4 is high-level diagram of a multicomputer SAR processor. A key component of multicomputers is the interconnection network. The design of the interconnection needs to make a careful balance of complexity against cost since the choice of network is crucial to the performance of the system.

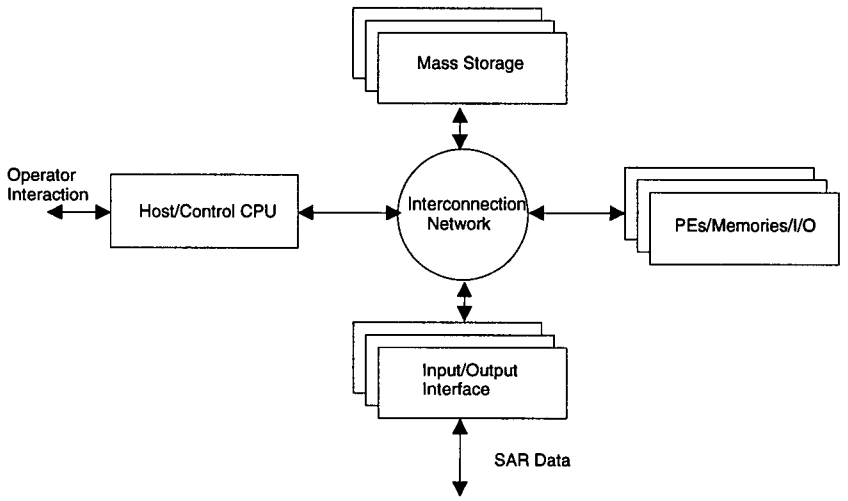


Figure 4.4: Typical multicomputer architecture

Some of the advantages of the multicomputer approach are listed below.

- Multicomputers are flexible. They can handle algorithmic changes as well as different algorithms.
- Either a complete off-the-shelf processor or off-the-shelf PEs can be used.
- Multicomputers are modular and easily expandable.

- Fault tolerance can be achieved.

The following is a list of some of the potential problems with multicomputers.

- The cost of the interconnection network can be significant.
- The inter-processor communication costs can severely hamper performance.
- Load balancing can be a difficult problem.
- Programming is more difficult than for shared memory machines.

Some examples of multicomputer SAR processors are the RSARP processor that is based on Mercury multi-i860 processor boards, the SAR processors based on Meiko Transputer based multicomputers, and the MDA IRIS X2C VASP based airborne processor which is a multicomputer-pipeline combination.

4.6 SIMD Processor

In a SIMD processor, a control unit issues the same instruction to a synchronous set of processors which each operate on a subset of the data. Figure 4.5 is a high-level diagram of a SIMD processor. One of the key design issues in SIMD architectures is the complexity of each PE. SIMD PEs can be simpler than those in multicomputers since they do not need to perform any program control functions. Many SIMD machines built to date have had large numbers of very simple PEs. This approach has not been well suited for long word length or floating point operations. The design of the interconnection network is another key issue and the same considerations described for multicomputers apply here.

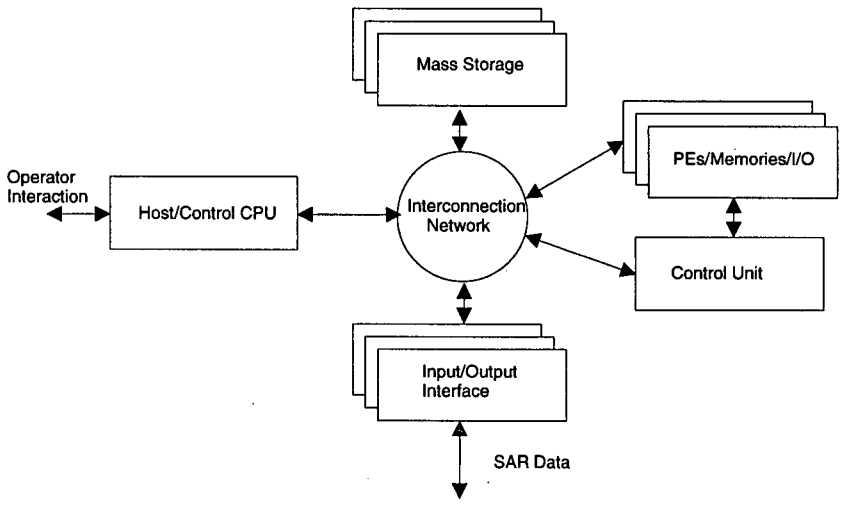


Figure 4.5: Typical SIMD architecture

The regular nature of the data processing operations in SAR point to SIMD processors as a possible solu-

tion. SIMD machines are well suited to FFTs and vector multiplies; however, interpolation operations can pose efficiency problems. Since SIMD machines are best suited to vector operations, a way of efficiently performing the algorithm's scalar computations needs to be found.

Some of the advantages of the SIMD approach are listed below.

- SIMD processors are generally more efficient for vector and FFT calculations than MIMD processors.
- Either a complete off-the-shelf processor or off-the-shelf PEs can be used.
- SIMD processors are modular and easily expandable.
- Fault tolerance can be achieved.

The following is a list of some of the potential problems with SIMD processors.

- It can be difficult to map an algorithm onto the SIMD architecture.
- Pipelining is not possible in a pure SIMD processor.
- SIMD processors can be inefficient for scalar operations, non-regular portions of SAR processing, and post-processing operations.
- The simple processing elements in SIMD machines often lack floating point support.
- Load balancing can be a problem. It may be difficult to keep all the PEs busy if the problem size changes.

Machines that use SIMD architectures have traditionally not been used for high performance SAR processing. However, numerous studies have been done to examine the feasibility of using SIMD machines for SAR. An example using a MasPar machine is described in [11] and another using a Thinking Machines CM-2 in [60].

4.7 Hardwired

Hardwired architectures are a general category for processor architectures that use custom application-specific designs and, therefore, do not fit into the other categories. Hardwired machines typically do not use regular von-Neumann processors. Rather, they use logic that is designed specifically for the application. An arbitrary amount of parallelism can be used. The hardware can also be tailored to the operations and wordlengths called for in the algorithm. Hardwired processors may be limited to a specific algorithm or they may be reconfigurable to some extent. Examples of this kind of architectures are processors implemented in discrete logic, custom VLSI, or programmable logic.

5 Computing Technology

This section gives a very brief summary of the key enabling technologies for SAR processing. A survey is made of the present state of the art and some indication is given of future directions. The technology areas covered are general and special purpose processors, programmable logic, memory, and interconnect.

The information summarized here is drawn from a wide range of sources. The contribution of this section is to pull together and organize the information that allows these technologies to be compared on the basis of their suitability for SAR processing.

In the sections that deal with processing devices, 1k CFFT times are used as a method of comparison. FFT times are not a good general benchmark: much better ones are available for general purpose microprocessors. However, this section looks at more than just general purpose processors. In order to process SAR data almost all processors will have to perform FFTs. As a result, FFTs are a reasonable common benchmark. The other performance measure that is used is OP/s. OP/s has become a relatively meaningless measure given the diversity in the definition of an operation. However in this section, the definition of OP is the one given in Section 2.5.1 and will usually be based on FFT or FIR times. The purpose of these performance measures is not to allow detailed comparisons between individual devices but rather to allow order of magnitude comparisons between different types of devices.

5.1 General Purpose Microprocessors

This section looks at some of the developments in general purpose microprocessors of both the RISC and CISC varieties. Table 5.1 gives approximate FFT times for some currently available high performance microprocessors. There is a wealth of information in the literature describing these chips. Much more information can be found on the internet [24].

It should be noted that many of the FFT times given in the table are for benchmarks coded in a high level language. Often significantly higher performance can be achieved using hand optimized assembly language. This should be kept in mind when comparing these processors to DSP chips, where benchmark results are almost always given for optimized assembly code.

Although many types of high performance microprocessors are available, some common trends have emerged. One such trend is the increased emphasis on floating point performance. Until a few years ago, floating point performance was a weak point with microprocessors, which often required a separate chip to provide hardware floating point support. The most recent chips have integrated floating point support that

Table 5.1: High performance microprocessors and their FFT performance

Manufacturer	Chip	1k CFFT				SPEC Benchmarks		
		Time (msec)	MOP/s	Clock (MHz)	Lang-uage	int92	fp92	Clock (MHz)
DEC	Alpha 21064	0.48	107	200		133	200	200
HP	PA-RISC 7100	1.1	47	99	C	80	150	100
IBM/Apple/Motorola	PowerPC 601	4.0	13		C	40	60	50
Intel	486DX	60	0.9	33	C	28	13	50
Intel	Pentium	1.25	41	133	assembly	67	64	66
Intel	i860	0.74	70	40	assembly			
MIPS	R4000	5.4	9	100	C	59	61	100
MIPS	R4400	3.3	16	150	C	88	97	150
Sun	SuperSPARC	0.95	54	50	assembly	89	103	60
Sun	UltraSPARC	0.3	171	167	assembly	275	305	167

sometimes surpasses their integer performance. This development has made it easier to implement general DSP applications on these chips.

The general purpose nature of these processors leads to some problems when they are used for DSP purposes. One of the problems with general purpose microprocessors in DSP applications is their extensive use of on-chip caching to attain high performance. The caching systems are designed for general applications like operating systems and are not appropriate for many DSP applications. Microprocessors also lack the built-in support for common DSP operations like FIR filters and FFTs that DSP chips have. However, recent advances in microprocessors have helped to minimize these differences. Now the FFT performance numbers attained by microprocessors are almost as good as those of the fastest general purpose DSP chips [82]. This can be seen from Table 5.1 where assembly language optimized FFT routines on the latest crop of RISC processors, like the DEC Alpha and the Sun UltraSPARC, show extremely high performance. The most important difference between the use of microprocessors and DSPs in DSP applications is, in many cases, no longer performance. The major differences are now found in other areas like power consumption, on-chip peripherals, inter-processor communications support, and cost.

One device that deserves special mention is the Intel i860. It was not used as a general purpose CPU but was targeted at signal processing type applications. This can be seen from its excellent FFT times. It found a great deal of use in multiple processor signal processing systems. Many different multi-i860 boards are available from a number of vendors. However, the DSP industry is looking at other chips for future growth since Intel does not plan any more developments with the i860 family.

Systems that make use of multiple standard microprocessors are becoming the most common kind of parallel machine. At the high end, supercomputer manufacturers are using large numbers of microprocessors in their systems. For more moderate performance, multiprocessor workstations are becoming very commonplace. There are also many board-level and subsystem products available that can be used in parallel systems.

5.2 General Purpose DSPs

One of the main drivers behind the rapid growth of the DSP market has been the availability of powerful low-cost DSP chips. Table 5.2 summarizes the currently available high-end DSP chips [58][19]. DSP chips can be divided into two types: floating point and fixed point. Floating point devices are easier to program but are more expensive. Fixed point devices are more difficult to program since truncation error needs to be carefully monitored by the programmer but they are less expensive and are the devices of choice in high-volume applications.

Table 5.2: High performance DSPs and their FFT performance

			1k CFFT			
Manufacturer	**Chip**	**Type**	**Time (msec)**	**MOP/s**	**Clock (MHz)**	**Notes**
Analog Devices	ADSP-21020	32-bit floating	0.58	88	33	
Analog Devices	ADSP-2106x (SHARC)	32-bit floating	0.46	111	40	multiprocessing support
AT&T	DSP32C	32-bit floating	2.8	18	50	
Motorola	DSP56002	24-bit fixed	0.83	62	80	
Motorola	DSP96002	32-bit floating	1.05	49	40	multiprocessing support
Texas Instruments	TMS320C40	32-bit floating	1.55	33	50	multiprocessing support
Texas Instruments	TMS320C80 MVP	32-bit fixed/floating multiprocessor				1 floating pt. & 4 fixed pt. processors
Zoran	ZR38001	20-bit fixed	0.67	76	33	

Currently the TMS320C40 and the ADSP-21060 SHARC are the most popular devices for high-end DSP applications. This is partly due to these chips' built-in support for multiprocessing. Both the C40 and the SHARC have six 8-bit communication ports that are intended to facilitate inter-processor communication. A number of manufacturers offer scalable board products with multiple C40s or SHARCs.

One of the most impressive chips to become available is the TMS320C80 MVP from Texas Instruments. This chip incorporates four integer processors and one floating point processor on one die. The peak performance is about 500 MOP/s. The MVP is primarily aimed at video signal processing applications like MPEG processing. This use of multiple processors on a chip can be seen as a sign of things to come.

While general purpose RISC microprocessors have caught up to DSPs in terms of arithmetic performance, DSPs still have a number of advantages. In contrast to RISC chips, DSPs often have separate instruction and data buses, multiple data buses, numerous on-chip peripherals like DMA controllers, inter-processor communications support, lower power consumption, and lower cost.

As always, the 1k CFFT performance figures give an incomplete indication of actual performance. As an example, Figure 5.1 shows the operation rate achieved on FFTs of various lengths by the C40, SHARC, and i860 processors. A sharp drop-off in performance is evident at the point where the FFT length exceeds the size of the on-chip memory for the C40 and i860.

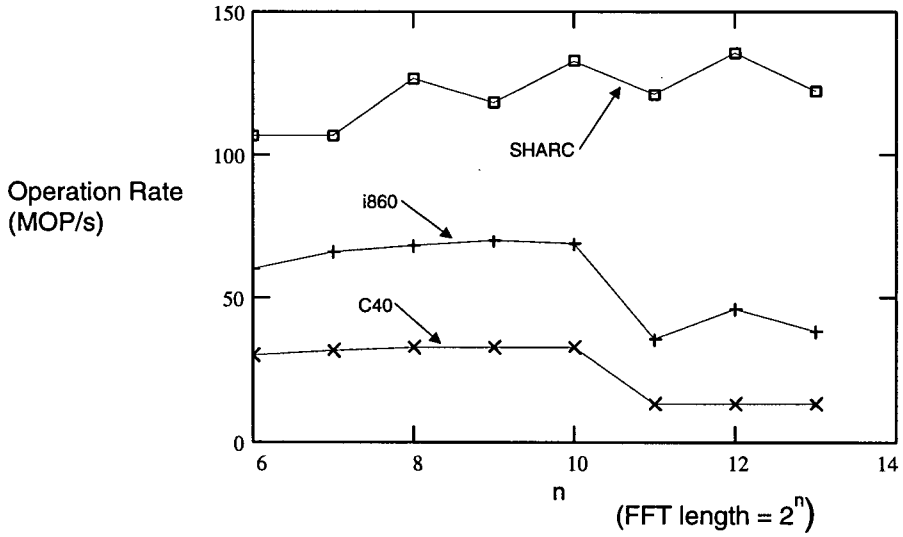


Figure 5.1: Performance for FFTs of various lengths

Many board level products that incorporate DSP chips are available. The boards typically contain from one to eight DSP chips and various I/O and memory configurations. A great deal of support software is also available, ranging from compilers to DSP operating systems.

5.3 Special Purpose DSPs

This section summarizes some of the currently available special purpose DSP processors. These devices are distinguished from the general processors described in the previous section by their promise of higher

performance for specific algorithms. Section 5.3.1 describes some accelerated FFT chips, and Section 5.3.2 lists some recent digital filter chips.

5.3.1 Accelerated FFT Chips

Since the FFT is such a commonly used DSP algorithm, a variety of special purpose chips have been developed to accelerate its execution. Table 5.3 lists some of the currently available products.

Table 5.3: FFT processor ICs

		1k CFFT					
Manufacturer	**Chip**	**Time (μsec)**	**MOP/s**	**Clock (MHz)**	**Data Format**	**Max FFT Size**	**Notes**
Array Microsystems	A66111 and A66211	131	391	40	16-bit floating	64k	supports 16 functions (but no FIRs)
GEC Plessey	PDSP16510	96	533	40	16-bit block floating	1024	FFTs only
LSI Logic	L64280	256	200	40	24-bit floating	2048	FFTs and MACs only
Raytheon/ TRW	TMC2310	514	100	20	16-bit fixed	1024	supports 16 functions
Sharp/Butterfly	LH9124 and LH9320	81	632	40	24-bit block floating	unlimited	supports 26 functions
Texas Memory Systems	TM66 swiFFT	76	674	40	32-bit IEEE floating	unlimited	supports basic complex arithmetic

The chips listed in the table have 1k CFFT execution times that are up to an order of magnitude faster than those of the fastest general purpose DSPs. In addition to execution time, a number of important features differentiate these chips. One of the most important of these is the generality of the device. Some perform only FFTs and nothing else. Others are more general purpose and support a set of common DSP operations. Some other characteristics are the number of chips required for a basic system, the maximum supported FFT length, the number format, and the wordlength. It is usually possible to use multiple chips to increase performance.

The most popular of the latest crop of these chips is the Sharp/Butterfly LH9124/LH9320 chip set. This chip set is near the lead in the race for the fastest single chip FFT time. It also supports a wide range of other arithmetic functions which is important for SAR processing since FFTs are not the only computationally expensive operation. The possibility of using it as the basis for a SAR processor architecture is examined in Section 9.3. The newer TM66 chip is similar to the LH9124 but is slightly faster and supports IEEE floating point arithmetic.

A number of manufacturers have board level products available that make use of accelerated FFT chips. Currently, the most common products are ISA or VME bus boards that contain single or multiple LH9124 chips.

5.3.2 Digital Filter Chips

Since digital filtering is the most basic operation in DSP, quite a few special purpose chips have been developed to perform fast filtering. Some of more recent ones are listed in Table 5.4. These chips are typically designed for applications that require extremely high-speed real-time filtering like video signal processing. These chips usually work with fixed point numbers with fairly short wordlengths and do not usually directly accommodate complex numbers. They do, however, achieve extremely high speeds in terms of numbers of operations per second.

Table 5.4: High speed filter ICs

Manufacturer	Chip	Data (bits)	Coef-ficient (bits)	Max. # of Taps	Approx Max. MOP/s	Max sample rate for N-tap FIR (MHz)	Max sample rate for 50-tap FIR (MHz)
GEC/Plessey	PDSP16256A	16	16	128	800	25/(N/16)	8
GrayChip	GC2011	24	14	32	4500	70/(N/32)	44
Harris	HSP43124	24	32	256	180	45/(N/2 + 1)	1.7
Raytheon/TRW	TMC2243	10	10	3	120	20/(N/3)	1.2
Zoran	ZR33288	8	10	288	8234	14.32 for all filter lengths	14.32
Motorola	DSP56200	16	24	256	19	10.25/(12+N)	0.165

Even though these chips have limitations, they do have potential applications in SAR processing. Section 9.5 looks at some possible applications of digital filter chips in SAR processing. Not many off-the-shelf products are available that make use of these devices since they are usually designed into custom systems.

5.4 Custom and Semi-Custom VLSI

The highest possible performance for a DSP algorithm that today's technology will allow can only be achieved by designing custom VLSI circuits to implement the algorithm. Custom VLSI chips are usually only used for extremely high volume applications or applications where specific power, space, or performance requirements dictate their use. VLSI design costs have been decreasing with the increased use of sophisticated CAD tools. The use of hardware description languages (HDLs) and synthesis tools has made it easier and much faster to develop complex VLSI designs. These advances have made VLSI and ASIC design an option for a much larger number of designers than in the past.

Custom VLSI design has not been a cost effective approach for general purpose SAR processors since these are only built in very low volumes. This is not likely to change in the near future despite the advances in VLSI design tools. One of the reasons custom designs are not as attractive for SAR processing is that the majority of the computation in SAR is made up of standard operations like FFTs for which standard parts are available. An exception to this may arise for SAR processors intended for airborne or spaceborne applications where stringent requirements rule out the use of general purpose parts.

In addition to fully custom VLSI, several types of semi-custom devices are available that allow the designer to customize only several mask layers of an otherwise standard chip. The lower density end of the ASIC market is being taken over by user programmable devices such as FPGAs.

5.5 Field Programmable Logic

Field programmable gate arrays (FPGAs) are user programmable integrated circuits that consist of uncommitted logic elements that can be interconnected in a general way. FPGAs have become extremely popular and, as a result, their prices have decreased rapidly. Great advances in density have also been made. Higher capacity FPGAs are announced almost weekly and 100 000 gate capacity FPGAs are expected to be available within the next two years [38].

Until recently, FPGAs lacked the density to support meaningful DSP applications. As FPGA densities increase, the interest in implementing DSP algorithms in FPGAs grows. FPGA based computers are discussed in Section 6.6.

5.6 Memory

The capacity of DRAM chips has been increasing dramatically since their introduction. The most common DRAM sizes available today are 1, 4 and 16 Mbits. It is expected that 64 Mbit chips will soon be commonly available. 256 Mbit chips are in development. Figure 5.2 plots DRAM size against the year of peak production for various DRAM chip sizes [76].

The ever increasing size of memory chips is very important to the design of SAR processors due to the large image sizes typically involved in SAR processing. For example, a 4k $\times$ 8k complex image, where each pixel is represented by two 32-bit complex numbers, requires 256 Mbytes of memory. Table 5.5 lists the number of memory chips needed to hold such an image.

Unfortunately, while DRAM capacities have increased greatly, access speeds have not increased by the same amount. Currently, memory speed poses the most serious bottleneck to increased processor perform-

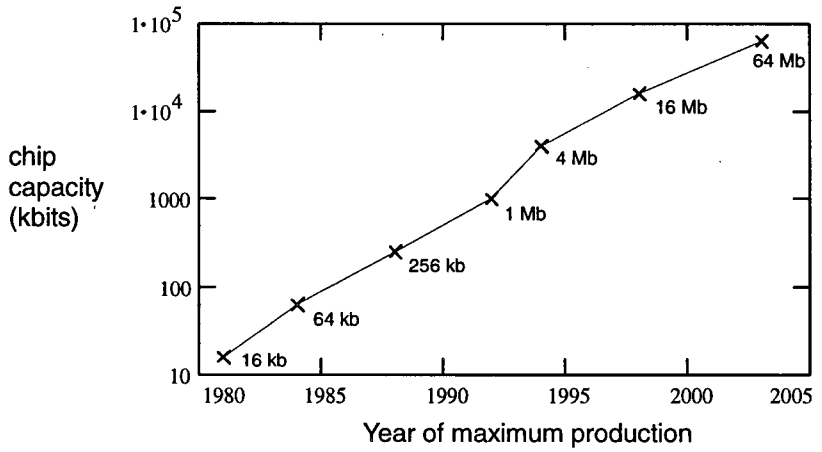


Figure 5.2: Memory chip sizes and their year of peak production

Table 5.5: Effect of memory chip size on chip and module count

Chip size (Mbits)	Number of chips for a 256 MB memory	Number of modules containing 16 chips for a 256 MB memory
1	2048	128
4	512	32
16	128	8
64	32	2
256	8	1

ance [14]. At present no absolute solution to this problem exists. Partial solutions are provided by caching and memory interleaving. Caching is the most commonly used scheme. In the current generation of microprocessors, elaborate caching schemes are used in an attempt to hide the slow speed of the DRAMs.

New developments in the DRAM devices themselves are also appearing to help alleviate the access time problem. These schemes usually allow the data to be accessed much more quickly if the data is accessed by successive memory locations. Some of the more common examples of these devices are listed in Table 5.6 in order of their date of availability [20]. DRAMs with fast page mode access are standard today. EDO DRAMs are expected to become commodity products in the near future. In several years, synchronous DRAMs are predicted to become the most common type of DRAM. It should be stressed that even though certain access sequences are greatly accelerated by these new DRAMs their random access time is still essentially unchanged.

The new types of DRAMs such as EDO DRAMs are potentially useful in SAR processing because much of the data accesses occur to sequential memory accesses. However, a problem arises when the data needs

Table 5.6: Bandwidth of some DRAM types

Type	Approximate Bandwidth (MHz)
Regular DRAMs	10
Fast page mode DRAMs	33
Extended Data Out (EDO) DRAMs	50
Burst EDO DRAMs	66
Synchronous DRAMs	100

to be accessed in a different direction, like during a corner turn in SAR processing. If these types of devices are to be used to increase performance, the implementation must take the characteristics of the devices in account when writing and reading values from memory.

5.7 Interconnect

In addition to large amounts of computing power, high performance processors require sufficient communications bandwidth. This bandwidth is provided by interconnection technologies. Like memory, currently available interconnections have not kept pace with the rapid growth in computing power. However, the primary reasons have not been technological. Rather, the problems have been to do with standards and market acceptance. Demand in the marketplace for interconnect has been concentrated on well accepted standards. This has resulted in interconnection technologies like the ISA bus remaining common long after higher performance buses became available.

In recent years, there have been many developments in interconnection networks. There has been a proliferation of standards and pseudo-standards. It is difficult to classify these interconnection schemes. The traditional boundaries between buses and I/O interfaces have become blurred. Serial interfaces are presently being used where previously parallel interfaces or backplane buses were used. This section presents one simple classification and gives examples of currently available interconnections. More information on modern interconnections can be found in [71], [70], and [51].

- Personal computer and workstation buses are interconnections that are commonly used in PCs and workstations as backplane buses. Table 5.7 summarizes the most common currently available buses.
- High speed buses and related interfaces are interconnections that are either backplane buses or additions to existing backplane buses. These interconnections are typically used in embedded or control applications. The most common category is the VMEbus and its derivatives. Table 5.8 lists the more common members of this class.

Table 5.7: Personal computer and workstation buses

Bus	Clock Rate (MHz)	Maximum Throughput (MB/s)	Typical Throughput (MB/s)	Max # Slots	Bus Width	Comment/ Applications
Industry Standard Architecture (ISA)	8.33	8.33	1-2	no logical limit, practically 3-8 from electrical characteristics	16-bit	similar to x86 bus, used in IBM PC compatibles
Extended Industry Standard Architecture (EISA)	8.33	33 in burst mode	8.33	logical limit 15	32-bit	used in IBM PC compatibles
Micro Channel Architecture (MCA)	10-20	157 in burst mode 76 sustained	30	15	16 & 32-bit	used in IBM PS/2
VESA Local Bus (VL)	up to 66	160 at 32 bits 267 at 64 bits (50 MHz)	67	3	32 or 64 bit	used in PC graphics/video applications
Peripheral Component Interconnect (PCI)	up to 33	132 at 32 bits 264 at 64 bits	65	10 loads, typically 2-3 cards	32 or 64-bit	used in PCs
NuBus	10	40	10	up to 8 typically	32-bit	developed by Apple
M Bus	40	320 in burst mode	200 sustained	6 theoretically, 4 in practice	64-bit	developed by Sparc vendors
S Bus	20-25	80 at 32 bits 168 at 64 bits		limited by electrical characteristics	32 or 64 bit	developed by Sun Microsystems,
Turbo Channel	12.5 - 25	50 to 98		2-6	32 bit	developed by DEC

- The other high speed interfaces category encompass new interconnection schemes used for interprocessor connections as well as processor to peripheral connections. Some of the high speed interfaces are listed in Table 5.9. The numerous new standards for lower speed interfaces are not listed.

The categories and tables listed above do not cover LAN standards or standards aimed primarily at peripheral interfaces like SCSI and its variants.

Existing high performance SAR processors have usually used some of the standardized interfaces in combination with a higher speed proprietary interface. Examples include the VASP signal bus from MDA or the interconnection network from Mercury. As with all areas of computing, there is a great desire to use standardized interfaces in the future. Some of the new high speed interfaces listed in the tables are aimed directly at high performance multiprocessor systems that can be used for applications like SAR processing. Examples include VME additions like Autobahn and RACEway.

Table 5.8: High speed buses and related interfaces

Interface	Parallel/ Serial	Topology	Maximum Throughput (MB/s)	Maximum # Slots	Bus Width	Comment/ Applications
Multibus I	parallel	bus	12	15	16 bits	
Multibus II	parallel	bus	80 in burst 64 sustained at 20 MHz	21	32 bits	
VME32	parallel	bus	40	21	32 bits	
VME Subsystem Bus (VSB)	parallel	bus	40	21	32 bits	sidebus to VME, uses P2 connector
VME64	parallel	bus	80	21	64 bits	extension to VME, uses address lines to transfer data
Futurebus+	parallel	bus	3.2 GB/s at 256 bits	14	32 to 256 bits	
AutoBahn	serial	user defined	200	>8	2 serial pins	extension to VME, developed by PEP Computer
Heterogeneous InterConnect (HIC) IEEE P1355	serial	user defined	160			
QuickRing	serial	ring	200	16 (in one ring)	6 bits	developed by National Semiconductor
RACEway	parallel	crossbar	320	16	32 bits	extension to VME, developed by Mercury
SkyChannel	parallel	bus or crossbar	320		64 bit	developed by SKY computer

Table 5.9: Other high speed interfaces

Interconnect	Parallel /Serial	Topology	Throughput	Number of Nodes	Addressing/ Data Paths	Comments/ Applications
Scalable Coherent Interface (SCI) ANSI/IEEE 1596	parallel	point to point	1 GB/s theoretical 125 MB/s with CMOS chipset 500 MB/s with GaAs chipset	64 k	64 bit address, up to 256 data bits	high perf. parallel systems
Fibre Channel ANSI X3T11	serial	point to point	up to 100 MB/s	16 million	24 bit ID fields	mass storage interfaces
High Performance Parallel Interface (HIPPI)	parallel	point to point	100 MB/s per cable	millions	32 bit data per cable	
OPTOBUS	serial	point to point	20 MB/s per fiber			developed by Motorola

6 Architecture Implementation Alternatives

This section looks at some alternatives for implementing SAR processors based on the architectures given in Section 4 and the technologies surveyed in Section 5. Only implementations that show promise for high performance processing are considered. The alternatives discussed here were chosen on the basis of being the most cost-effective ones given the state of technology now and in the near future.

The implementation approaches considered in this section are listed below in approximate order of generality. The architectures (as defined in Section 4) that each approach encompasses is shown in brackets.

- Workstation (SP, MP)
- Accelerated general purpose computer (CN)
- Supercomputer (any)
- Network of workstations (MC) *
- DSP uniprocessor and multicomputer (SP, MC) *
- Reconfigurable computing machine (HW) *
- Custom algorithm specific processor (HW, PL or MC)

Each of these alternatives is discussed in the sections that follow. The options marked with asterisks are identified for more in-depth study in Section 9 based on the selection criteria presented in Section 3.5. The discussion in this section extends and updates previous work to encompass a wider range of alternatives. No broad ranging overview of the suitability of architectures for SAR processing, like the one presented here, has been found in the literature.

6.1 Workstation

The rapidly increasing performance of RISC based workstations has made them suitable for a wide variety of tasks that previously required large computers with special accelerators. For cases where a single processor is not sufficient, workstation vendors provide an upgrade path by allowing extra processors to be added. These parallel systems are multiprocessors (the processors access a shared memory). The large market for workstations gives them an advantage over other high performance systems, in that they are usually quick to incorporate new technologies and software advances. Although workstations can be used for DSP purposes, they are really optimized for running applications under general purpose operating sys-

tems like UNIX. This leads to a number of potential problems when they are applied to demanding DSP applications:

- Workstations make extensive use of caching. Depending on how the caching is handled, it is not necessarily useful and may even be detrimental for SAR processing (see Section 7.4.2).
- The shared memory and bus can be a major bottleneck in multiprocessor systems, especially for corner turn operations.
- Performance is difficult to predict in comparison with more specialized architectures.
- Workstations are slower at FFTs by a factor of 10 in comparison with the fastest FFT processor chips.
- Workstations are not suited for embedded applications like airborne or spaceborne processors.

More and more applications will move to general purpose workstations as their computational requirements are overtaken by the ever increasing capabilities of workstations. The flexibility, ease of use, and economies of scale that characterize workstations and PCs can more than make up for any extra costs associated with using a general purpose computer for an application. Unfortunately, the SAR processing requirements considered in this thesis are too demanding for cost effective implementation on single and multiprocessor workstations. SAR processing implementations on general purpose workstations are not looked at further in this thesis.

6.2 Accelerated General Purpose Computer

Accelerated general purpose computers have been a very common implementation architecture for SAR processors. These systems are a type of common node architecture and the comments made in Section 4.2 about common node architectures also apply here. Typically, a powerful superminicomputer like an Alliant or a DEC VAX is used as the common node. It is supported by an attached accelerator which can be composed of an array processor from a vendor like STAR or Sky, or custom accelerator boards. Today the accelerator is likely to be a multicomputer composed of DSP chips. An advantage of accelerated general purpose computers is that most of the components are likely to be available off-the-shelf. Most of the algorithm is usually implemented in software on the central computer so the processor is relatively easy to develop and modify. The main disadvantages of these systems are their cost and the performance limitations due to the bottleneck posed by the common node. Some analysis and some examples of accelerated general purpose computers are given in Section 9.3.3 of [27]. Since this approach does not scale well to higher performance levels, it will not be examined further in this thesis.

6.3 Supercomputer

The traditional solution to large scale scientific computation has been the use of supercomputers. A wide variety of architectures have been used to build supercomputer-class machines. Some examples are vector supercomputers like the Cray series, MIMD machines like the Thinking Machines CM-5 and the Kendall Square Research series, and SIMD machines like the Thinking Machines CM-2 and the Maspar series.

The execution of SAR algorithms on supercomputers has been examined in quite a few instances [60] [11] [66]. Traditionally, however, supercomputers have not presented a cost effective solution for SAR processing. This is primarily due to their high cost. Also most high performance SAR processors are production processors that execute an application that is limited to a number of common DSP algorithms. Supercomputers attempt to provide a high degree of generality. This is not needed for most SAR processors but contributes greatly to the system cost.

Recently traditional supercomputers have become an even less attractive option. Currently there is a trend away from such systems. This trend has been demonstrated in the recent bankruptcies of many supercomputer manufactures like Cray Computer, Thinking Machines, and Kendall Square Research. At one point, massively parallel processor (MPP) machines were thought to be the wave of the future. However, these have also not done as well as expected. This was due to high prices, long and costly development times, and the difficulty of software development.

Traditional supercomputers are not examined in any more detail in this work. However, some of the lower cost supercomputer replacements that many researchers are investigating for some applications are considered. Some examples are networks of workstations, multiprocessor workstations, and reconfigurable computing.

6.4 Network of Workstations

A new option for computationally intensive problems is the use of networked groups of the many powerful and inexpensive workstations that are available today. This idea is rapidly growing in popularity and is attracting much research interest [5]. Some of the factors that have contributed to usefulness of networks of workstations (NOW) are listed below.

- New technologies tend to appear in workstations faster than in larger systems like supercomputers.
- Workstations provide more computing power at a lower price than most other computing systems.

- Excellent software is available for workstations since the high cost of software development can be spread over large sales volumes.
- An opportunity exists to use the unused capacity of existing workstations for distributed tasks. This essentially provides computing power at no additional cost.
- Unused RAM capacity of existing workstations can be put to use for applications like distributed disk caches.

Several inter-process communication standards for high performance computing on workstation clusters have been established. The two most popular ones both have freely available software distributions: parallel virtual machine (PVM) [42], and message passing interface (MPI) [31]. Both of these systems provide software that facilitate message passing interfaces between programs running on different workstations [46]. The performance figures that are currently available show that relatively high overheads can be expected. For example, in an MPI performance evaluation paper, an FDDI network capable of 100 Mbps was used, but MPI could only reach a maximum throughput of under 20 Mbps [73]. As the software matures, it will no doubt be tuned to lower some of the overheads.

For SAR processing applications, workstation clusters have all the advantages and disadvantages of using single workstations (Section 6.1). NOW attempt to circumvent as many of the disadvantages as possible by using more and more workstations in parallel. An examination of the literature shows that while these systems are being used for a large number of applications, these applications are mostly ones that were previously implemented on supercomputers. Little experience exists in predicting the usefulness of NOW for high performance SAR applications. (A PVM based system using older technology is described in [25]).

The main disadvantages of existing NOW systems are the severe performance limitations due to the high latency and low bandwidth of their communication networks. Huge overheads imposed by the existing software packages that support NOW have magnified this problem. In order for NOW implementations to be successful, both a high bandwidth network and low overhead communications are necessary. NOW-like supercomputers, such as the IBM SP and the Cray T3D/E, overcome some of these limitations but at increased cost. The development of the Scalable Coherent Interface (SCI) [44] also attempts to address these issues.

The key to successful algorithm implementation on workstation clusters is to come up with a partitioning of the problem that has a computation to communication mix that matches that of the workstations and their interconnection network. Since SAR processing can be split into fairly large chunks of computation it

appears to be a promising application for workstation clusters. This option is examined further in Section 9.1.

6.5 DSP Uniprocessor and Multicomputer

Currently the most common approach to building high performance DSP systems is to use DSP ICs as the building blocks. General purpose, special purpose, or even custom DSP chips are used in single or multiple processor configurations. The key characteristic of this approach is that the architectures use components optimized for DSP operations but are flexible enough to handle a wide range of algorithms.

The multicomputer architecture with general purpose DSPs as the processing elements combines the advantages of multicomputers with those of DSP chips. Modern DSP chips like the TMS320C40, TMS320C80, DSP96002, and ADSP-21060 have on-chip support for multiprocessing. This usually consists of high bandwidth parallel data ports with associated controllers that allow interprocessor communication to occur with minimal overhead to the main CPUs. Depending on how the processors are interconnected, these systems can be quite scalable. Hundreds of DSP chips can used in parallel.

Many commercial products are available in the area of DSP based systems. A great deal of software is available to support development. This includes compilers, debuggers and operating systems. However, the software support for parallel systems is not yet as mature as for uniprocessor systems. A major disadvantage of current DSP based systems is that there are not many industry wide standards for hardware or software. Each chip, board, and software vendor has a different set of standards. This makes development more difficult and severely hampers portability.

It is no surprise that DSPs are being used for many of the SAR processors designed recently. Some examples of SAR processors based on general purpose DSP based multicomputers include MDA processors based on the MDA/Spectrum VASP multi-96000 boards, and on Mercury Computer's multi-i860 boards.

The RASSP program in the United States has defined a series of benchmark projects that will allow the development process for application specific signal processors to be studied, measured, and refined. The first two of these benchmarks involve the design of a real-time airborne SAR processor [95]. Two contractors designed systems to meet the requirements set by the RASSP program. Both teams chose a multiple processor DSP chip based approach [2]. One team designed custom boards and used a specialized DSP chip: the LH9124. The other team used commercial boards that feature either the ADSP-21060 or the i860. The choice of these processors is hardly surprising since each of these chips has the fastest 1k FFT time for the processors currently available in its class as shown in Tables 5.1, 5.2, and 5.3.

The DSP chip approach to building SAR processors is studied in much more detail in Section 9. Section 9.2 looks at processors based on general purpose DSPs. Vector DSP based architectures are covered in Section 9.3. Section 9.4 discusses optimized DSP ICs. Processor designs based on digital filter chips are examined in Section 9.5. Fine grained parallel approaches based on DSPs are discussed in Section 9.6. Finally, a sample architecture that combines vector and scalar DSPs is described in Section 9.7.

6.6 Reconfigurable Computing Machine

Reconfigurable computing machines use reprogrammable building blocks to implement application specific processors. Such computing devices have been given a number of different names: reconfigurable computers, transformable computers, virtual computers, or flexible processors. All of these names refer to the concept of using reconfigurable or programmable hardware to implement processing devices. These processors attempt to use hardware to circumvent the one-instruction-at-a-time bottleneck inherent in Von Neumann computers. Reconfigurable computing is a rapidly growing research area [54][17][7][23][55].

FPGAs are usually used as the reconfigurable building blocks in such machines. The reprogrammability of FPGAs allows algorithm implementations to be debugged and modified like software but still run at the speed of dedicated hardware. The major problems with FPGAs in high performance processing applications are the low speed and density of the existing FPGA devices. FPGA gate density is typically an order of magnitude below that of microprocessors. As a result, unless the algorithms to be implemented are very simple, multiple FPGA devices are needed.

Some examples of reconfigurable computing machines based on multiple FPGAs are the Programmable Active Memory (PAM) developed by the DEC Paris Research Lab (PRL) [10], the Anyboard system developed at North Carolina State University [88], the SPLASH system from the Supercomputing Research Center [40][89], and the processors developed by Virtual Computer Corp. [54][22]. An up-to-date and extensive list of FPGA based computing machines is available on the internet [74]. The key feature of all of these machines is the ability to repeatedly prototype and optimize designs quickly and cheaply. The growing popularity of these machines has led Xilinx, the leading FPGA vendor, to recently announce a new family of FPGAs, the XC6000 family, that is optimized for reconfigurable architectures.

Reconfigurable computers have been used for a variety of applications but until recently little work was done in applying them to DSP. This is changing quickly and a flood of FPGA applications in DSP is beginning to appear [6][48][49][39]. In instances where FPGA based computers were used for DSP, the algorithms have typically been fairly simple and involved short wordlengths.

A comparison of FPGA custom computers and conventional processors in DSP applications is given in [9]. It is concluded that, at present, reconfigurable architectures do not give any cost/performance improvements over conventional uniprocessor and parallel processor systems for DSP applications. This is mainly because commercial chips have dedicated hardware support for arithmetic. Custom computers offer the best speed-ups when they can implement in hardware what sequential computers have to do in software. The extensive on-chip support for fixed and floating point arithmetic in modern DSPs and general purpose CPUs makes them difficult to surpass using programmable logic.

Two major challenges are evident in all the FPGA based reconfigurable systems developed to date. The first is the relatively low density of the FPGA devices. This requires that large arrays of FPGAs be used. The limited number and low speed of the off-chip connections make such designs difficult and time-consuming. The second challenge is the need for a new set of tools to deal with the design of reconfigurable machines. At the higher level, a new type of compiler is necessary to specify programmable hardware designs. At the lower level, better synthesis and place-and-route tools are required.

Since there is an overhead associated with reprogrammability, FPGAs will always be a step behind single purpose VLSI devices for fixed function DSP applications. For example, it will always be possible to design an FFT processor in a custom VLSI chip that is faster than a version implemented in FPGAs. However, since FPGAs have the advantage that one device can be used to implement more than one algorithm, they may be the more cost-effective solution for some applications.

The applicability of reconfigurable computing to SAR processing is examined in more depth in Section 9.8.

6.7 Custom Algorithm Specific Processor

Custom processors are the traditional approach to building very high throughput DSP systems. Custom processors are defined to be processors that are hardwired for one algorithm and cannot easily be used for another algorithm that is significantly different. Custom processors make use of either custom or commercial VLSI chips on custom circuit boards.

This type of processor can utilize an arbitrary amount of parallelism and can achieve extremely high performance. Custom SAR processors have usually used some form of coarse grain pipelining. For example, the NASA Advanced Digital SAR Processor (ADSP) built by JPL implements the range-Doppler algorithm in a hardwired pipeline [27]. It achieves a computation rate of over 6 GFLOPs at a cost of 73 boards. A simplified and more modern design is the NASA Alaska SAR Processor (ASP). It achieves about 3

GFLOPs at a cost of 35 boards. The IRIS airborne SAR processor from MDA is also a custom hardwired pipeline processor.

The cost-effectiveness of custom SAR processors needs to be examined careful before embarking on a custom processor design. The major disadvantages of custom processors are high development cost, poor flexibility, and potentially low reliability. These factors rule out the use of custom processors in many applications. In addition, the speed of commercial more general purpose architectures has increased greatly, and these architectures are now suitable for applications that previously required a custom approach. As a result, there is a trend today away from custom processors. Custom processors are not considered further in this thesis since one of the goals of this work is to find high-performance architectures that are also flexible.

7 Memory System Design

SAR processing requires extremely large amounts of memory and high data transfer rates. It is important to emphasize memory system design right from the beginning of the design process since slow DRAM access times or memory access conflicts can easily make memory latency the main bottleneck in the system. SAR system designers have an advantage over general purpose computer system designers since the memory access patterns in SAR can be largely predicted. This predictability must be exploited in the design of high performance but cost effective memory systems. Memory requirements also set SAR processors apart from many other high performance DSP systems. Typical high throughput DSP systems are designed to perform a single set of operations on a stream of data. They do not have the memory support to perform repeated operations on a large data array that is stored in memory and accessed in various directions.

This section examines the various issues involved in memory systems design for SAR processing. There is no reference that explains the design process for such systems. The contribution of this section is to draw from the wealth of well-known design ideas related to memory system design, choose the most important ones, and apply them to SAR processing.

High performance data path design is introduced in Section 7.1. Section 7.2 looks at the memory access patterns in SAR. The problem of corner turning is discussed in Section 7.3. The most common ways of overcoming the problem of high memory latency are covered in Section 7.4.

7.1 High Performance Data Path Design

The general goal of high performance data path design is to minimize latency and maximize throughput. Many different factors can cause these areas to miss their performance goals. Latency problems can be caused by slow RAM access times, cache misses, and memory or bus contention. Throughput problems can stem from slow memories, slow clock speeds, narrow data path widths, and interference from other tasks or processors. These problems can also cause the latency and throughput to be non-deterministic, which is a problem in real-time systems. In a real-time processor, it should be possible to exactly predict all data transfer times.

Some general data path design techniques that can help to achieve high performance are listed below.

- Use multiple data buses to increase throughput and minimize contention.
- Dedicate each data bus to a small number of precisely defined uses.

- Use as wide a word width as possible on each data path.
- Use high data transfer clock rates.
- Operate the data buses synchronously where possible to avoid synchronization delays.
- Use a fast, deterministic bus arbitration method.
- Isolate data paths with double buffers or dual port memories.
- Use one of the techniques from Section 7.4 to reduce average memory latency to the required level.

An appropriate balance between performance and cost must be found when choosing which of these techniques to apply in a system.

7.2 Memory Access in SAR Processing

In SAR processing, memory is generally accessed in a regular fashion along the rows and columns of the data array. This suggests that RAMs which provide faster access for consecutive addresses would be useful. Once a row or column has been read from main memory, it is typically used a number of times before the next row or column is required. This indicates that caching of data rows or columns may be used to advantage.

The main exception to the rule of regular accesses is the corner turn operation, where memory accesses switch between row-wise and column-wise. This point in SAR algorithms was a major bottleneck in the days when the data was stored on disk. One would expect that once the entire data array is stored in RAM, the need for corner turning would disappear. However, the relatively slow random access time of DRAM devices and the characteristics of the methods that are used to overcome this DRAM latency problem have made this untrue.

7.3 Corner turns

7.3.1 Singe Processor Corner Turns

Whether or not a corner turn is necessary depends on several factors:

- **Caches**

  The presence and organization of caches will often be the deciding factor in the decision to perform corner turns. Caches are discussed in more detail in Section 7.4.2.

- **RAM access speeds**

  The difference in RAM cycle times between random access and sequential access will help decide if a separate corner turn is worthwhile.

- **Memory access patterns**

  In a system with RAMs that have faster sequential access, the number of times that memory is accessed in the orthogonal direction between implied corner turns will help determine if a separate corner turn operation is worthwhile.

The effect of the last two factors is examined in more detail in Section 7.4.4.

The simplest way of performing a corner turn is to simply read one row and column, and then exchange them while writing them back to memory. This requires temporary storage of the size of one row or column. In this method, one half of the memory accesses will be in the non-preferred direction.

A number of methods exist for performing corner turns in cases when memory can only be read in one direction [32][33]. Most of these methods were developed for data arrays that were stored on disk or tape. They require multiple passes through the data matrix. In general, the size of the temporary storage area is traded-off against the number of passes that are needed through the data array. Whether or not these methods offer a performance improvement in the case of DRAM memory depends on the difference in access times for random and sequential access.

If it is decided not to perform a corner turn, it is still possible to perform 2D processing with only single direction accesses. One method for performing the 2D FFT with accesses only in one direction is described in [4]. This approach is potentially useful for range-Doppler SAR processing since it simplifies the handling of RCMC.

7.3.2 Multiple Processor Corner Turns

A corner turn in a parallel system involves an exchange of data between PEs such that each PE obtains a strip of data that is orthogonal to the one it had originally. The corner turn is troublesome in parallel systems since it involves only communication between memory and processors and no computation. This stresses the interconnection bandwidth in the parallel system. Unfortunately in many parallel systems, communication is much slower than computation.

The communication-only nature of corner turns can expose bottlenecks in parallel systems. For example, on a system with a shared memory or a shared bus, the parallel speed-up can be very low during corner

turns. This is due to the fact that all PEs attempt to access the shared resource at the same time. For corner turning to be truly parallelized, distributed memories and parallel communication paths are necessary.

In a multicomputer with a horizontal partitioning of the SAR algorithm, the need for corner turns depends on the way the data is partitioned. If the data is suitably overlapped, each processor only needs to operate on one portion of the data (static partitioning) and a global corner turn operation can be avoided. In this case, each PE may not need to communicate with the other PEs during processing. However, this approach leads to inefficiencies and inaccuracies due to the required data overlaps.

The decision whether or not to perform global corner turns during processing depends on the relative costs of the corner turn and the decrease in efficiency that results if no corner turn is done. The decrease in efficiency in the no corner turn case arises from data overlaps due to filter lengths. The cost of a corner turn depends on the speed of interprocessor communication and the topology of the interconnection network.

In a vertically partitioned (pipelined) system, the corner turn memory is usually used as the memory buffer between pipeline stages. The corner turn buffer can be single or double buffered. Concurrent access to both rows and columns from two pipeline stages can be achieved in both cases.

The simplest situation is the use of full double buffered memories. This has the advantage of having simpler control sequences and the ability to access any part of the entire image. However, it has twice the memory cost of the single buffered scheme.

A single buffered corner turn memory is less expensive but requires a somewhat more complicated control and addressing mechanism. The control logic has to ensure that no access conflicts occur. Also the output data from a single buffered corner turn memory is only available in a staggered format.

7.4 Memory Latency

Although SAR processing has different characteristics than many other processing problems, the ways around the memory latency problem are the standard methods of

- wide data path widths,
- caches,
- interleaving, and
- faster RAMs.

7.4.1 Data Path Width

SAR processing typically operates on fairly wide data words. The most common arrangement is to use complex 32-bit numbers for a total of 64 bits. The transfer of one complex word would take eight cycles on an 8-bit data bus. Obviously it is desirable to use a data bus that can accommodate the full data word width.

7.4.2 Caches

Caches are extremely widely used in general purpose computers. Caches typically operate by fetching a whole data line from main memory when one address is requested. This is very inefficient if the memory accesses are orthogonal to the direction of the cache lines. This is exactly what can happen during a corner turn operation. In the worst case, each memory access will result in a cache miss and a whole new cache line will need to be fetched from memory. As a result, on many modern general purpose computers, an explicit corner turn operation is required to make memory accesses efficient in both directions.

Some other considerations on the use of caches in SAR systems are given below.

- Data caches can also cause problems if they are not large enough to hold the entire quantum of data being processed: a range or azimuth line in the case of SAR. If a cache is too small, a data line may have to be fetched from main memory multiple times instead of just once.
- While normal data caching schemes can be a problem in SAR processing, program code caching can be quite useful in SAR since signal processing code is often quite small. The compute intensive portions often consist of several tight loops which can be stored in a cache.
- Caches are especially problematic in a real-time system since caching can introduce non-deterministic delays.
- If possible, it can be advantageous to disable the caching mechanism during some parts of processing.

Most of SAR processing operates on one range or azimuth line at a time. This indicates that a cache that holds one or several lines would be useful. A data line can be fetched from main memory and stored in a high speed cache memory. If the cache is made double buffered, one of the buffers can be written back to memory and refilled with the next data line while the other one is being processed. Provided the memory system can keep up with the processor, memory latency and data transfer times can be completely hidden in this way. The access time requirements of the cache memories can be reduced by using separate caches for the input and output to the processor. Figure 7.1 shows a sample architecture that makes use of these techniques.

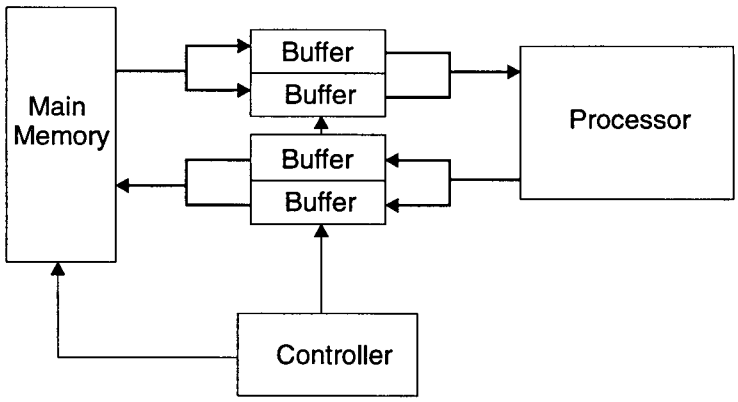


Figure 7.1: Sample double buffered caching scheme

7.4.3 Interleaving

Interleaved memory banks can be used to improve the average memory access time. Memory locations are spread out among a number of parallel memory banks. When an address is activated, a location in each bank is accessed simultaneously. Provided all these locations need to be accessed by the processor, the effective memory access time is reduced to t_{access}/N, where N is the number of memory banks. Memory interleaving also has some disadvantages. Interleaving requires a more complicated memory controller. It also requires that memory be upgraded in larger increments since the size of each memory bank must be increased together.

Like the other schemes for speeding up memory access, interleaving relies on the fact that most memory accesses are to sequential locations. Thus it should be possible to design an effective memory interleaving scheme for SAR processing since the memory access patterns can be predicted. In SAR, fast sequential access is required to both the rows and columns of the data array. The techniques developed for parallel memory access in array processors [18][57] are directly applicable here. In these methods, there is a general trade-off between the complexity of the memory system and its potential performance. More sophisticated schemes require more elaborate memory controllers and alignment networks. However, since memory access methods in SAR processing are confined to row and column accesses, a straightforward skewed storage approach can be used to provide fast access in both the row and column directions. More complex methods, such as prime memories, are not needed.

Figure 7.2 illustrates the storage of a data array with no interleaving, skewed interleaving, and a simpler interleaving scheme. With four memory banks, skewed storage gives a four times improvement in memory access time. However, skewed storage requires a fairly complicated memory control system. The second

interleaving scheme shown in Figure 7.2 only provides a two times improvement in memory access. However, it is much simpler since it relies on the fact that the array sizes are a power of two and several of the address lines can be used to rearrange the data in memory. Through the use of a programmable memory controller, various array sizes can be accommodated. If the array dimensions are not powers of two, there will be some wasted storage space since this scheme will need to use the next largest power of two.

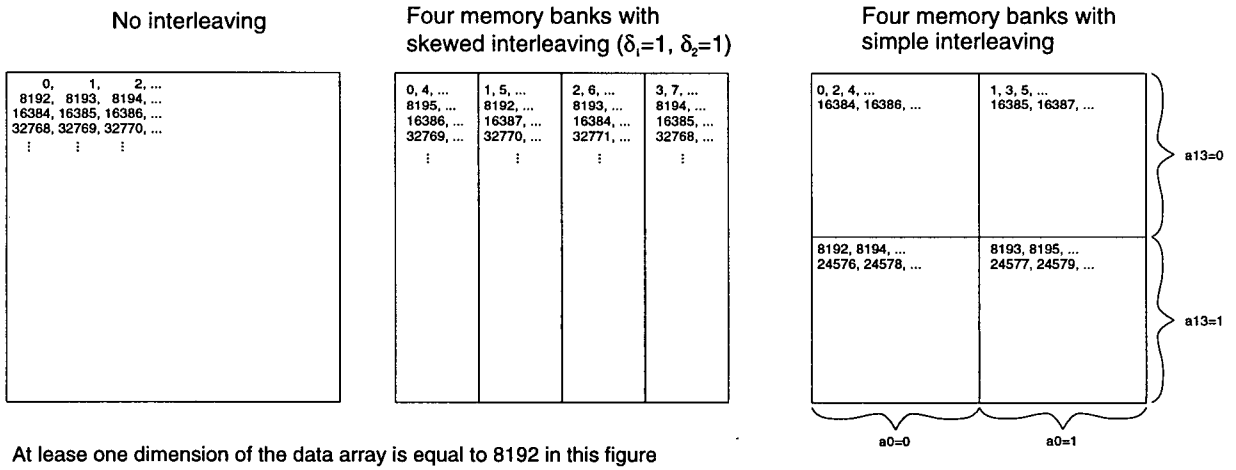


Figure 7.2: Some interleaved memory schemes

7.4.4 Fast DRAMs

Most new families of fast DRAM devices provide accelerated access to sequential memory locations but do not improve the random access time. In SAR processing, the memory must be accessed sequentially in orthogonal directions. This causes problems with the use of the new fast types of DRAMS like burst, column, nibble, EDO, and synchronous. All of these devices allow fast access only in one direction. In order to reap the benefits of this fast access, the data would have to be explicitly corner turned during execution. This would allow all accesses for processing to occur at the fast speed. However, during corner turning, either some of the accesses have to occur at the slow random access speed of the DRAM or several passes through the data must be made.

Figure 7.3 shows a graph of the average access times for an 8k × 8k data array for regular DRAMs and fast DRAMs as a function of the ratio of number of accesses to number of corner turns. In the case of the regular DRAMs, no corner turns are performed and all the accesses occur at the slower random access speed. In the case of the fast DRAMs, two corner turns are performed and all other accesses occur at the fast speed. It is assumed that the random access cycle time is 110 ns for both types of DRAM. The cycle time for the fast DRAM is assumed to be 40 ns in fast direction.

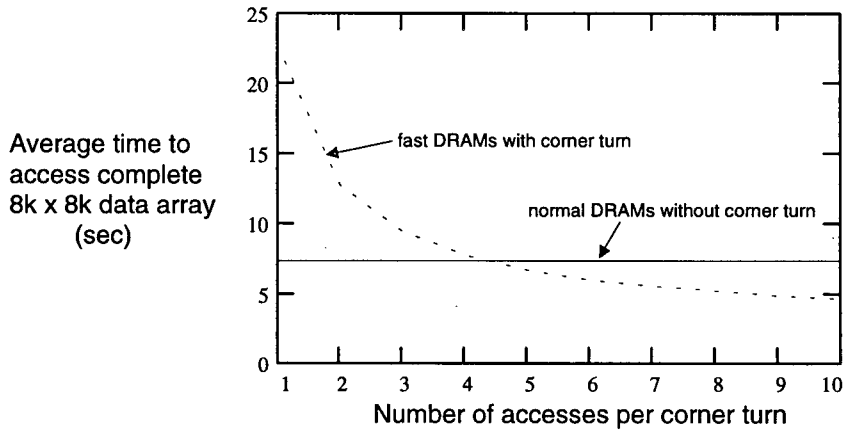


Figure 7.3: Average data array access time in slow direction when using fast RAMs

The figure shows that the choice of memory configurations depends on the number of times the data is accessed between corner turns. In SAR this depends on both the algorithm and the architecture and can range between two and ten. The graph shows that the crossover point for this scenario is about four accesses per corner turn. This can be reduced by using a more sophisticated corner turning scheme like the one described in [32] at the cost of a larger temporary storage area. The crossover point can be reduced to two or three accesses if a temporary storage array large enough to hold 32 rows is used.

Other types of fast DRAMs incorporate on-chip caches with extremely fast cache to memory transfers. These chips do not necessarily offer good performance for SAR because of the fixed mapping and relatively small size of the caches. This includes the new DRAMs that adhere to the RAMbus standard. These chips have extremely fast peak transfer rates (up to 500 MB/s) but have the dual disadvantages of fixed caches and a narrow data path (8 bits).

Since SAR processors use large amounts of memory, it is desirable to use readily available inexpensive DRAMs as opposed to more expensive SRAMs. At present, this means fast page mode DRAMs and in the future it may mean EDO or synchronous RAMs. In any case, the overall design of the SAR system should attempt to take advantage of the characteristics of the DRAM devices used in the memory.

8 Interconnection Networks

Interconnection networks are a key architectural component of most parallel systems. This section looks at some of the issues related to interconnection networks for modern parallel SAR systems. The focus is on those networks that are the most cost-effective with today's technology. The networks are analyzed with respect to the requirements of SAR processing. In particular, an original analysis based on corner turn and matrix transpose performance is presented.

There are two facets to interconnections in parallel systems:

- system I/O and
- inter-processor communication.

These two areas are examined in the sections that follow.

8.1 System I/O

System I/O is comprised of the data transfers necessary to move the data into and out of the SAR processor. Since these transfers often occur over a global system bus, they are a potential bottleneck. The delay due to system I/O can be removed by double buffering the input and output at the cost of additional memory. The cost of this option is often prohibitive in SAR systems due to the large image sizes. System I/O interconnection should offer

- high bandwidth,
- independent I/O channels,
- efficient interfaces to mass storage devices,
- minimal requirements for CPU intervention, and
- double buffering where necessary.

Since the design of the system I/O interconnection is usually driven by the requirements of the specific system, it is not explored in more detail here.

8.2 Inter-processor Communication

The inter-processor connection is responsible for carrying the communication that occurs between the PEs. Some of the key attributes required of interconnection networks for SAR processors are a high data trans-

fer rate, scalability, and efficient support for corner turning. The greatest test of the interconnection network in a SAR processor arises during parallel corner turn operations.

The discussion presented here focuses on the network topology. As with all aspects of the interconnection, the performance and cost of a given topology depends largely on the type of inter-processor communication support built into the PEs. There are many possible interconnection networks topologies [35]. Some common topologies that are practical for use with current DSP chips are the bus, mesh, and crossbar. These topologies are examined in the sections that follow.

8.2.1 Bus

The bus is perhaps the simplest interconnection scheme as can be seen from Figure 8.1. It can be implemented with almost any DSP chip. DSP chips that have multiple independent buses have a definite advantage for bus interconnections since they can separate the local and global memory accesses onto distinct buses. The TMS320C40 and the DSP96000 are examples of DSP chips with independent buses.

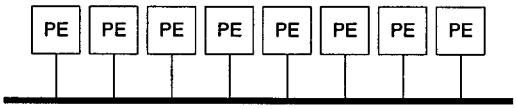


Figure 8.1: Bus interconnection

Two problems with buses are poor scalability and slow corner turns. Some of these disadvantages can be overcome by using multiple buses within the system. This is the approach taken by the VASP boards from Spectrum Signal Processing. These boards use multiple independent high bandwidth buses in conjunction with two-port memories to provide multiple data paths.

During a global corner turn in a distributed memory system, each PE must communicate with all other PEs. This is problematic on a bus since all data transfers must be serialized. The time required to perform a corner turn on a system with a single bus can be expressed by

$$N_{pe} \cdot (N_{pe} - 1) \cdot \frac{N_r \cdot N_a}{N_{pe}} \cdot \frac{1}{N_{pe}} \cdot 2 \cdot 4 \cdot \frac{1}{BW}, \tag{8.1}$$

where

- N_{pe} is the number of processors,
- N_r is the dimension of the data array in range,

- N_a is the dimension of the data array in azimuth, and
- BW is the bandwidth of the bus in MB/s.

Complex 32-bit values are assumed. Figure 8.2 shows a plot of the time required to corner turn an 8k × 4k image on a system with a bus bandwidth of 150 MB/s against the number of PEs.

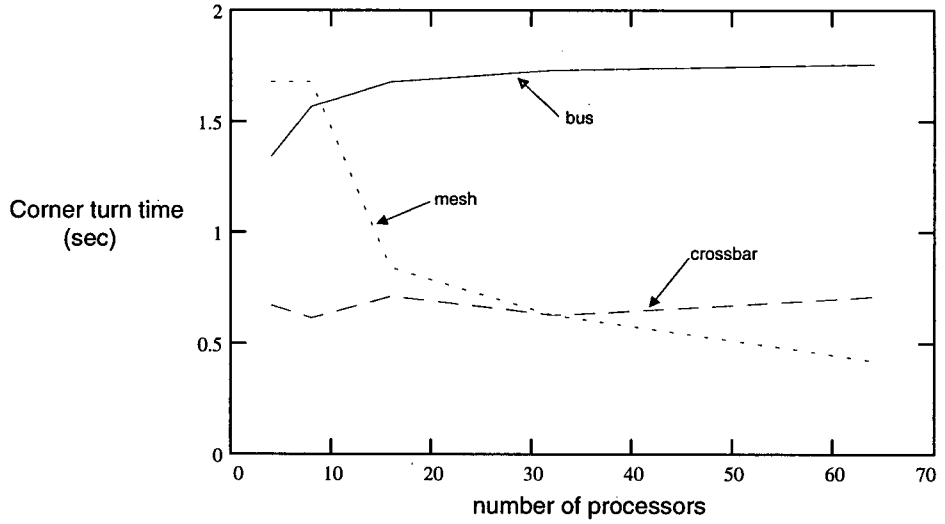


Figure 8.2: 8k × 4k image corner turning time on various interconnection networks

8.2.2 Mesh

Meshes are the simplest two dimensional interconnection scheme. Figure 8.3 shows a sample of a mesh interconnection.

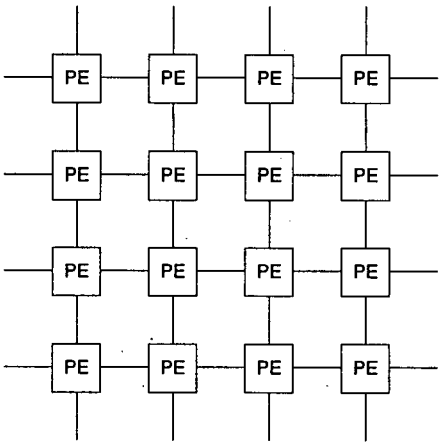


Figure 8.3: Mesh interconnection

Meshes are typically implemented using DSP chips that have on-chip I/O ports designed to support multi-processor communications. Both the TMS320C40 and ADSP21060 have six such communication ports.

The on-chip ports allow for very low chip count solutions. For example, the ADSP21060 with its 512 kB of on-chip memory can form a single chip PE. Board products that support meshes of DSPs are available from Ariel, Spectrum, Integrated Computing Engines, and numerous others.

Meshes have the advantage of being quite efficient at corner turns. The time required for a corner turn on a mesh of processors can be expressed as:

$$\left\lceil \frac{\sqrt{N_{pe}}}{2} \right\rceil \cdot 2 \cdot 4 \cdot \frac{N_r \cdot N_a}{N_{pe}} \cdot \frac{1}{BW}, \tag{8.2}$$

where the symbols have the same meanings as in Equation 8.1. A toroidal mesh is assumed. This implies a maximum data shift distance of $\left\lceil \left(\sqrt{N_{pe}} \right) / 2 \right\rceil$ hops. This expression is plotted in Figure 8.2 for an $8k \times 4k$ image with a 40 MB/s transfer rate between PEs.

Some disadvantages of meshes are as follows:

- The greater number of inter-PE connections makes them more difficult to build.
- The lack of direct access to each PE can make external I/O and broadcasts inefficient.

Hypercubes are a related type of interconnect. They are not as commonly used since they require more connections per node and since they do not scale as easily. Hypercubes exhibit similar corner turning performance to meshes. The corner turning time can be found from equation 8.2 by replacing the factor $\left\lceil \left(\sqrt{N_{pe}} \right) / 2 \right\rceil$ with $log_2 (N_{pe})$.

8.2.3 Crossbar

Crossbar interconnections offer the highest connectivity and also the highest cost. Crossbars have typically not been used in large multicomputers since full crossbar interconnections become very expensive. A commercial partial crossbar mechanism that circumvents this problem is the RACEway interconnect from Mercury Computer Systems [50]. Mercury's RACEway is a scalable crossbar scheme that has been standardized. RACEway uses custom six port crossbar switches that are capable of two simultaneous 150 MB/s transfers at a time. RACEway offers good scalability because interconnect performance and cost is not fixed like on a bus, but scales with the number of processes. Figure 8.4 shows some topologies based on RACEway-style crossbar switches.

Crossbar interconnects are often used for CPUs that do not have dedicated ports for interprocessor communication like the i860. Crossbar based board products using the i860 are available from Mercury, Sky, and CSPI.

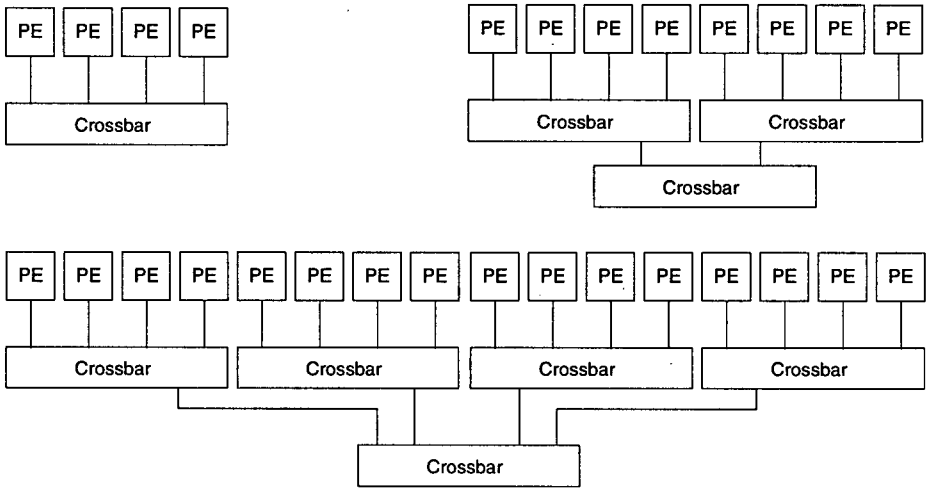


Figure 8.4: Crossbar interconnection

The time required to perform a corner turn on RACEway-style crossbar interconnections like the ones shown in Figure 8.4 can be expressed as

$$\left(\sum_{k=1}^{\lceil log_4(N_{pe}) \rceil} N_{pe} \cdot 3 \cdot 4^{k-1} \cdot \frac{4^k}{2 \cdot N_{pe}} \right) \cdot \frac{N_r \cdot N_a}{N_{pe}} \cdot 2 \cdot 4 \cdot \frac{1}{N_{pe}} \cdot \frac{1}{BW}, \tag{8.3}$$

where the symbols have the same meanings as in Equation 8.1. Since the RACEway crossbar chips support six ports, extra crossbars can be added to increase bandwidth and to provide fault tolerance. The model given here assumes no extra crossbars are used, which means at least one port is left unused in each crossbar chip. The corner turn time for an 8k × 4k complex data array is plotted in Figure 8.2.

Some of the issues that affect crossbar based systems are listed below:

- The cost of the crossbar chips is a major factor in these systems.
- The crossbar chips are often single sourced custom chips that increase the chip and pin count of the processing boards.

8.2.4 Conclusion

This section has looked at three common topologies. The mesh topology was found to have the most scalable corner turn performance. Many other possibilities exist that may also offer good performance. The corner turning performance of the topologies has been emphasized. This may not always be an important consideration. If the data is partitioned differently or if a global memory buffer is used, corner turning may not be necessary at all.

9 Examination of Architectures

This section brings together the trio of algorithm, architecture and technology that have been looked at separately in previous sections. The following design approaches are considered.

- network of workstations
- general purpose DSP architecture
- vector DSP architecture
- optimized DSP architecture
- digital filter chip architecture
- fine grained parallel architecture
- a sample vector/scalar architecture
- FPGA computing machine

The goal is to find architectures that meet the requirements set out in Section 3.5. This entails the use of the best currently available technology in cost effective architectures that are tuned to the needs of SAR processing but are still flexible enough to handle a range of algorithms.

The selection and analysis of the architectures in this section is new work. The results and conclusions presented here are new contributions to the study of SAR processor architectures.

9.1 Network of Workstations

9.1.1 NOW Model

This section presents a simple model that will be used to test the potential usefulness of NOW for SAR processing. The intent is to obtain an idea of the performance that could be expected and the factors that affect performance. Static horizontal partitioning is used as the partitioning approach since it involves the least inter-processor communication of all the partitioning strategies. The details of the model are given below.

- A network of ten workstations is assumed, where each workstation has either 25, 50, 75, or 100 MOP/s of sustained processing speed for SAR operations. These are reasonable values for current and near-future workstations. A 1 ms 1k CFFT time is typical for many modern workstations, which cor-

responds to 50 MOP/s. Ten 50 MOP/s workstations would be capable of approximately 1/10 real-time processing. As an example, a DEC Alpha workstation has a peak performance of 200 MOP/s and achieves about 100 MOP/s on a 1k CFFT.

- The processing scenario is that of the range-Doppler algorithm modelled in Section 2.5.4. The real-time computation rate is about 4.5 GOP/s. The image size is 160 MB at the beginning of execution and 140 MB at the end.
- All data is assumed to originate from a single node in the network and the output data is collected at the same node (see Figure 9.1). This is a realistic situation since a single frame synchronizer or tape drive is usually the source of the data. The network bandwidth figures used in this section refer to the bandwidth required at the connection(s) of the I/O node to the network. No other assumptions about the network topology are made.

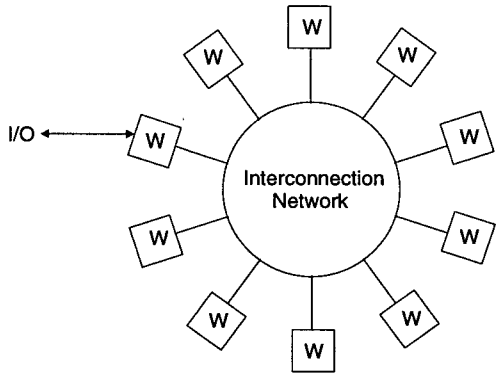
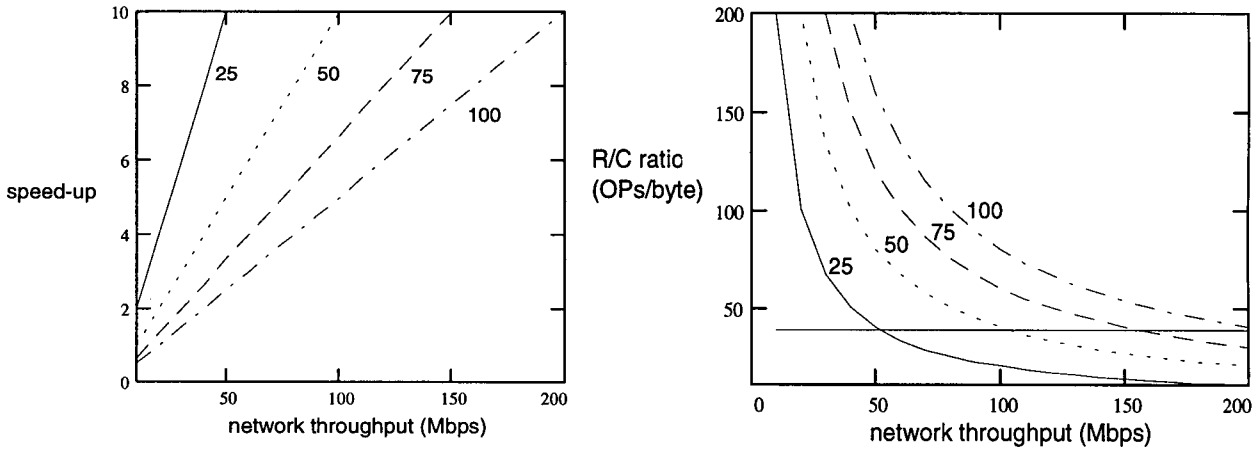


Figure 9.1: NOW SAR processor

- The data is partitioned between the processors at the beginning of the computation and is collected at the end. Thus, the only inter-processor communication is the transferring of the complete image data array twice: once at the beginning and once at the end.
- It is assumed that communication can be completely overlapped with computation. This implies that the execution time will be determined by the longer of the computation time and the communication time.
- Algorithm overheads due to things like data overlaps are ignored in this section. This allows the effect of network communications to be studied in isolation. It should be remembered that the actual speed-up may be considerably less depending on the algorithm parameters.



(numbers on graphs indicate speed of workstations in MOP/s)

Figure 9.2: Speed-up for ten workstation network vs network throughput and computation/ communication ratio vs network throughput

9.1.2 Results

The plot of speed-up versus network throughput in Figure 9.2 shows that the network bandwidth must be very large before the full computational power of the workstations can be realized. The speed-up is defined as the execution time on one workstation divided by the execution time on ten workstations of the same type. For example, an attempt to process a full size SAR image on a network of workstations connected by a conventional 10 Mbps Ethernet network would be a disaster: the speed-up would be less than one.

The relationship between computational performance and communications bandwidth can be described by the computation to communication ratio (R/C ratio). For example, a workstation capable of 100 MOP/s and a network capable of 10 MB/s have an R/C ratio of 10 OPs of computation per byte of communication. The SAR processing scenario used in this section has an R/C ratio of about 38 OPs/byte. If the R/C ratio of the architecture is lower than that of the algorithm, then the processors will be the bottleneck. If it is higher, then communication will be the bottleneck. R/C ratios are plotted as a function of network bandwidth in the second graph in Figure 9.2. A horizontal line indicates the ratio that characterizes the hypothetical SAR problem. The intersections between the curves and the line correspond to the communications throughputs at which the ten workstation network achieves linear speed-up.

Since the R/C ratio of the SAR problem is known, the network bandwidth required for a given amount of processing power can be predicted. This is shown in Figure 9.3, which plots the required communications bandwidth against the number of workstations. It can be seen that extremely high network bandwidths are required to achieve linear speed-up.

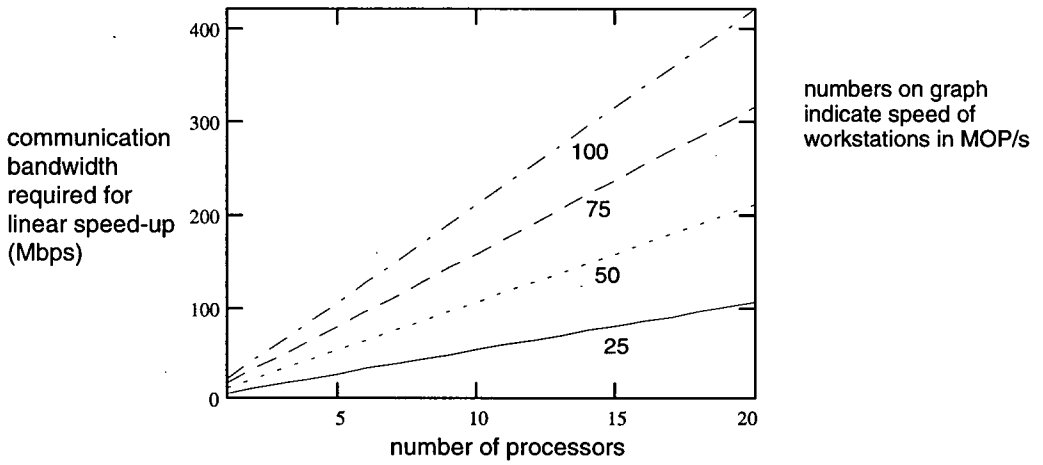


Figure 9.3: Network throughput required vs. number of workstations

If the more realistic assumption is made that only 50% of the computation and communication can be overlapped in time, then it becomes much more difficult to achieve linear speed-up. This can be seen from Figure 9.4 which is a plot of the speed-up versus network bandwidth for this new assumption.

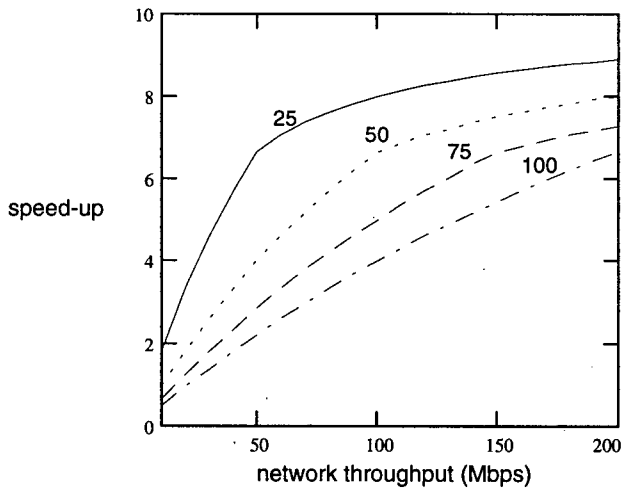


Figure 9.4: Speed-up for ten workstation network vs. network bandwidth with partially overlapped communications

9.1.3 NOW Conclusions

A performance analysis of the range-Doppler algorithm on a NOW architecture has shown that very large network bandwidths are necessary for practical implementation of SAR processing on NOW. For example, in a ten workstation network where each workstation has a performance of 100 MOP/s, an effective network bandwidth of over 200 Mbps is required. This is beyond most commonly available networking technologies. Extremely low overhead communications software is also a necessity. It should be stressed that

the bandwidth numbers shown in this section are effective numbers after all overheads. Protocol and operating system overheads can easily add 100% to the bandwidth required.

In the analysis of the NOW architecture, the horizontal partitioning approach was used since it requires the minimum amount of inter-processor communication. However, as was seen in Section 2.6.4, this approach also suffers serious efficiency problems due to data overlap requirements. An approach that includes a global corner turn could potentially solve this problem but it would increase interprocessor communication significantly.

NOW become more appropriate for SAR processing if the algorithm requires a significantly higher amount of computation per image pixel than the standard image formation algorithms like range-Doppler. A possible candidate application is a SAR processor that incorporates large amounts of computation in post-processing functions such as those required for image analysis and understanding.

9.2 General Purpose DSP Architecture

The most general purpose architecture that is still DSP-specific is one that uses general purpose DSP chips as building blocks. This section discusses the processor, memory, and interconnection subsystems for general purpose DSP based SAR processors. The performance of a sample architecture for SAR processing is also examined.

9.2.1 Processor

Some desirable features of a general purpose DSP for a SAR processing system are listed below.

- floating point number support
- especially fast FFT execution time
- high data transfer bandwidth
- multiple independent data buses
- inter-processor communication support
- large on-chip memory

Currently the chips that come closest to meeting this ideal are the Analog Devices ADSP-21060 SHARC [1] [16], the TI TMS320C40 [84], or the TI TMS320C80 [86]. These processors each have strengths and weaknesses. Before choosing any one processor, a trade-off analysis must be performed.

9.2.2 Memory

General issues related to memory system design were covered in Section 7. A few of the issues that are particularly important in DSP based designs are given below.

- the distribution of the total memory into local (to each DSP) and global sections
- the scalability of the memories over a large range of sizes
- the need for distributed memories for high bandwidth portions of the memory system
- the potential bottleneck posed by shared memory areas
- the careful use of caches or buffers
- efficient support for DMA transfers

9.2.3 Interconnect

The properties of several of the most common interconnection networks were discussed in Section 8. Some issues specific to DSP based systems are listed below.

- The choice of interconnection is affected by the inter-processor communications support provided by the DSP chip.
- The use of standardized interfaces makes the system more general and more easily expandable.

9.2.4 Sample Architecture

An example of a general purpose DSP based architecture is presented in this section. The goal is to design the most cost effective architecture that is capable of 1/10 real-time satellite SAR processing.

The AD21060 SHARC is selected as the DSP for the architecture. Some of the strengths of this chip are given below.

- large high bandwidth dual-ported SRAM (512kB)
- flexible DMA structure
- six 40 MB/s link ports
- high FFT performance
- IEEE floating point number format
- potentially a single chip PE for multiprocessor systems

The SHARC also has some weaknesses:

- only one external bus
- on-chip RAM may be too small for some applications

If the TMS320C40 were used instead of the SHARC, some design differences would result.

- The lower FFT performance would mean that more PEs are required.
- Each PE would require more external memory.
- The dual external buses would provide more design flexibility. The additional bus could be used as a global high bandwidth communication channel.

A toroidal mesh is chosen as the interconnection network for the processor. A block diagram of the processor is given in Figure 9.5. The mesh is supported by the SHARC with no need for extra chips. Two of the SHARC's link ports are left unused by the inter-processor connections. On certain PEs, one of these ports is used for system I/O.

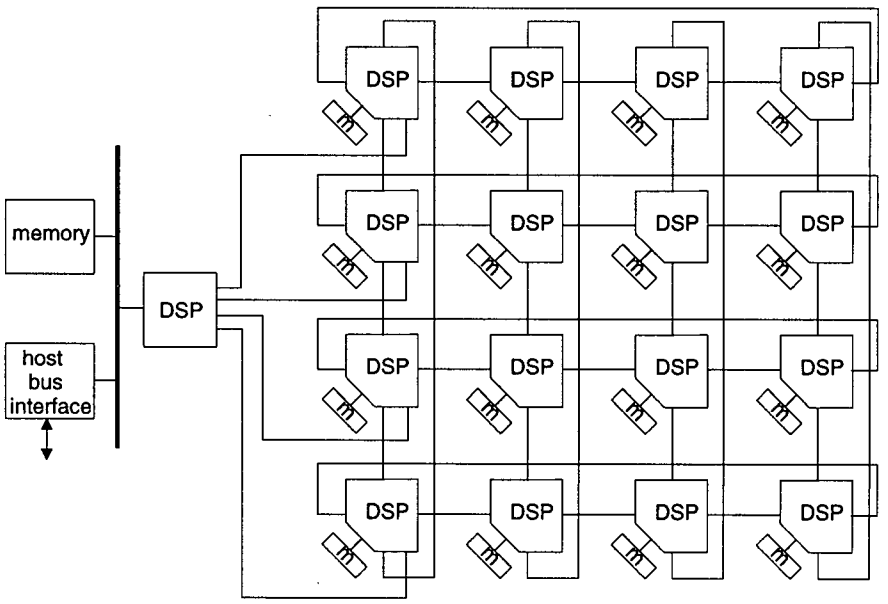


Figure 9.5: General purpose DSP based mesh processor

If it is assumed that the SHARCs can sustain 25% of the performance they exhibit on FFTs, then about 16 processors would be needed for 1/10 real-time satellite SAR processing. Since the SHARCs' internal memories are not big enough to hold the entire image, external memories are required at each PE. To process an 8k × 4k image, 16 MB of memory is required at each node.

All inter-processor communication occurs over the link ports. One of the link ports of each of the PEs at one edge of the mesh is connected to an additional SHARC. These four 40 MB/s links provide a 160 MB/s system I/O link. If more I/O capacity is required, an additional SHARC can be added and attached to the extra link ports of other PEs in the array. The other unused link ports can be used for additional I/O or for fault tolerance purposes.

A simple model was created to predict the performance of this architecture for the range-Doppler processing scenario of Section 2.5.4. The model shows that this architecture is capable of about 20% of real-time operation. The model takes into account the time to load data into and out of the mesh, all core SAR processing calculations as well as resampling, and the time required for two corner turns. Theoretical SHARC execution times are used. The time required by overhead functions executed by the DSPs and any delays due to external memory access have not been taken into account. Also, coefficient and filter calculation times have not been considered. These functions are fairly easy to accommodate in this architecture since they can be performed by any PE. However, they will lead to additional computation and communication.

Since it is reasonable to plan for about a 100% overhead to the theoretical model, a 16 processor architecture seems well suited to achieve 1/10 real-time operation. This verifies the earlier prediction based on FFT performance alone. In an actual design, it is often desirable to have a safety margin. If a 50% safety margin is used, then 24 processors are required.

9.2.5 General Purpose DSP Conclusions

General purpose DSP based architectures pose a good solution for low to medium speed SAR processing. They strike a reasonable balance between performance and flexibility. Compared to architectures based on microprocessors, they offer a lower chip count and lower cost solution. However, they are more specialized and require more design effort than SAR processors based on standard microprocessor systems. For processing speeds that approach real-time, very large numbers of processors are required compared to designs based on more specialized chips. The design of general purpose DSP systems is also somewhat riskier than designs of more specialized architectures since performance prediction is much more difficult.

9.3 Vector DSP Architecture

This section examines the design of a SAR processor based on the Sharp/Butterfly LH9124 vector DSP. This chip was chosen for a number of reasons: extremely fast FFT time, support for other operations besides FFTs, availability of boards and software, and adequate numerical precision (24-bit block floating

point). The time spent looking at LH9124 designs in this section is justifiable since the chip has a well thought-out architecture that is a good example of high performance design. The results obtained are also applicable to architectures based on other chips.

An overview of the LH9124 is given in Section 9.3.1. Standard single chip architectures are presented in Section 9.3.2. A mapping of the range-Doppler algorithm to an LH9124 based processor and some performance predictions are given in Section 9.3.3. Sections 9.3.4 and 9.3.5 give some multi-chip architectures and range-Doppler performance predictions. The performance of the LH9124 on some other SAR algorithms is discussed in Section 9.3.6. Section 9.3.7 gives some conclusions regarding the use of the LH9124 in SAR processing.

9.3.1 Introduction to the LH9124

The LH9124 is a high-performance fixed-point DSP optimized for vector oriented operations [80]. An overview of the chip is shown in Figure 9.6. Some of its key features are given below:

- The data word width can be 8 to 24 bit real or complex with support for block floating point.
- 25, 40 and 50 MHz versions are available and 66 and 80 MHz versions are planned.
- Data I/O occurs via four independent, bi-directional 24-bit complex buses.
- Built in operations are available for 26 functions like real and complex FIRs, multiplications, and radix-2, -4, and -16 butterflies.
- The heavily pipelined architecture is very efficient for long vector lengths.
- Programming is relatively straight-forward due to the high level of the instructions provided.
- Performance prediction is also quite easy.

Some shortcomings of the LH9124 are listed below:

- There is no on-chip RAM or address generator. Address patterns can be generated by a companion address generator chip: the LH9320.
- It is not suited for scalar computations.
- There are no comparative, branching, or flow control capabilities.
- The block floating point format is incompatible with floating point formats used by most general purpose DSPs.
- FIR filtering speed is much slower than the speed of specialized digital filter chips.

- A relatively large number of support chips, like address generators and SRAMs, are required.
- The interface to a larger memory system most likely requires a custom design.

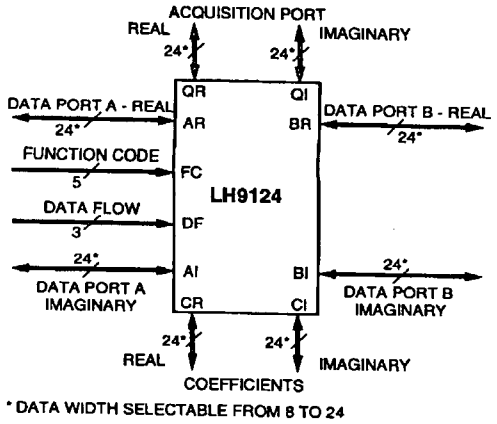


Figure 9.6: LH9124 digital signal processor

A list of execution times for the basic DSP operations is given in Table 9.1. FFTs are performed in passes composed of radix 2, 4, or 16 stages. For example, the most efficient way of computing a 1k FFT is to decompose it into two radix-16 stages and one radix-4 stage. The data vector is transferred through the chip for each pass. The highest processing efficiency is achieved with radix-16 butterflies.

Table 9.1: LH9124 execution times

Operation	Time	Time at 40 MHz (t_{cycle}=25 ns) (µsec)
Filtering	$(N_{filter} \cdot N + 18) \cdot t_{cycle}$ for each filtered point	-
Multiplication	$N \cdot t_{cycle}$	-
512 CFFT	$(3 \cdot 512 + 2 \cdot 68 + 18) \cdot t_{cycle}$	42.3
1k CFFT	$(3 \cdot 1024 + 2 \cdot 68 + 18) \cdot t_{cycle}$	80.7
2k CFFT	$(4 \cdot 2048 + 2 \cdot 68 + 2 \cdot 18) \cdot t_{cycle}$	209.1
4k CFFT	$(3 \cdot 4096 + 3 \cdot 68) \cdot t_{cycle}$	312.3
8k CFFT	$(4 \cdot 8192 + 3 \cdot 68 + 18) \cdot t_{cycle}$	824.8

Board products that incorporate the LH9124 are available from Catalina, Interactive Circuits and Systems, Pacific Cyber/Metrix, Valley Technologies, and others.

9.3.2 Single LH9124 Architectures

A sample single LH9124 configuration is shown in Figure 9.7. A double buffer is used on the A, B and C ports and a single buffer on the Q port. Any other buffering configuration is possible. Not shown in the figure are

- the scheduler that provides the overall control functions,
- the main memory, its controller, and its interconnection with the buffers, and
- a general purpose CPU for scalar computation.

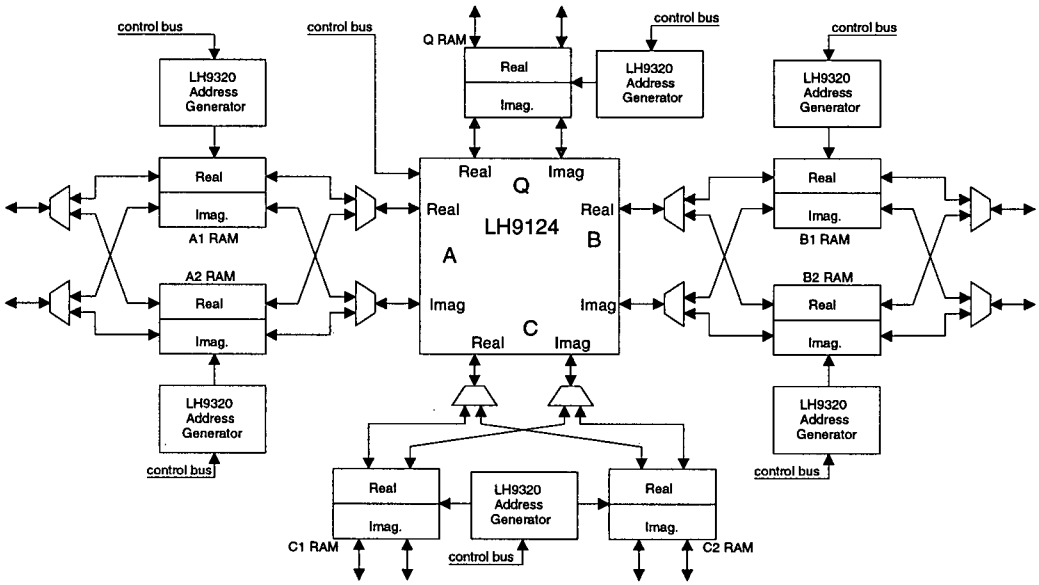


Figure 9.7: LH9124 single chip architecture

The CPU used for scalar computation must have access to the data being processed. This CPU could be a general purpose DSP, and will most likely be a floating point processor since it will perform more complicated higher level functions.

For the algorithms that are modelled in the sections that follow, it is assumed that all the other elements of the system are fast enough to allow the LH9124 to execute at its maximum rate. This is not unreasonable since most of the computation will be done by the LH9124. It will be shown that the data transfer rates are also not unreasonable provided the data path widths are wide enough to handle the wide word widths demanded by the LH9124. The task of keeping the LH9124 busy can be achieved by using:

- a main memory that is fast enough to fill and empty the buffers on time,
- double buffering to hide the time taken by data transfers, and

- a sufficiently fast scalar CPU.

Some of the design issues for the memory subsystem are

- the number of memories and memory buses,
- the nature of the connection between the LH9124 buffers and the main memory,
- the design and complexity of the control structure, and
- the connection point for external I/O.

Some sample interconnections between LH9124 based PEs and a main memory are shown in Figure 9.8. As always, a trade-off exists in these designs between complexity and performance.

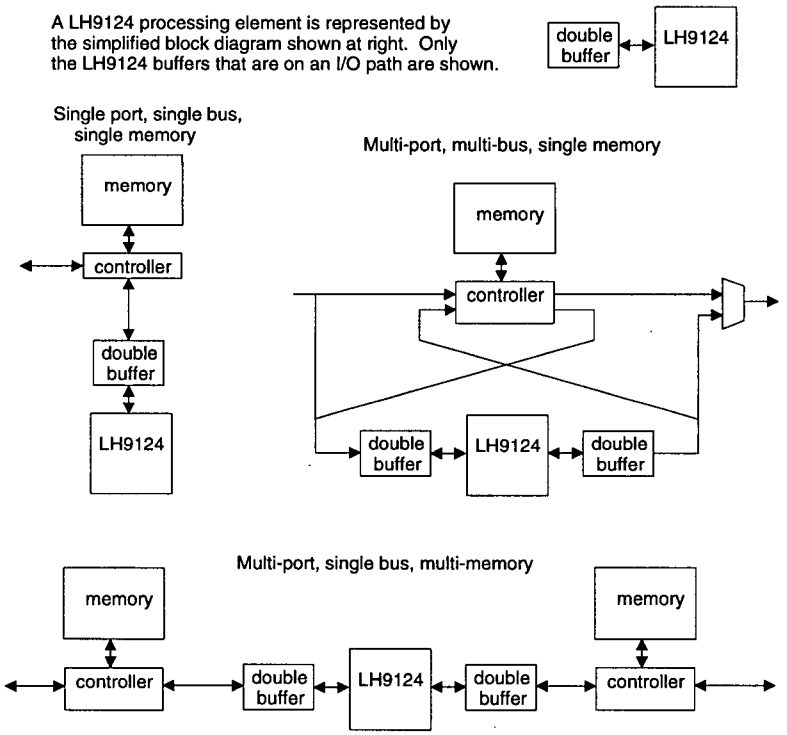


Figure 9.8: Some sample LH9124 to memory interconnections

The interconnection with scalar CPUs has not yet been addressed. There are many ways to do this and the method selected will depend on the specific application. This is explored in more depth in Section 9.7, where a design that combines the LH9124 with several scalar DSPs is presented.

9.3.3 Range-Doppler Algorithm on a Single LH9124 Architecture

An implementation of the range-Doppler algorithm described in Section 2.5.4 on an LH9124 is modelled in this section. The vector operations in the algorithm are assigned to the LH9124 and the scalar operations to the general purpose CPU. Table 9.2 gives the mapping for each step in the algorithm. The table also shows the execution times and data transfer rates for each of the steps allocated to the LH9124. Some assumptions that were used when generating these values are listed below.

- Complex vector multiplies are rolled into the first stage of the FFTs.
- A single bus, single main memory architecture is assumed (see Figure 9.8).
- The address generators are able to support all the operations required.
- A double buffer is used between the LH9124 and main memory. All data transfers between main memory and the double buffer occur while the LH9124 is processing a vector in the other half of the double buffer.
- The data is transferred back to main memory after the azimuth FFT and then fetched back in the range direction for RCMC. This essentially adds two corner turns, but the alternative of performing RCMC across multiple lines stored in the buffer is not supported by the LH9320.

Table 9.2: Range-Doppler algorithm implementation with a single LH9124

Algorithm step	Processor	Execution time (sec.)	Data transfer rate between main memory and buffer (MCW/s)
1.0 I/Q Balance	GP CPU	-	-
2.0 Range FFT, 3.0 Range Multiply, 4.0 Range IFFT	LH9124	5.8	9.9
5.0 Azimuth FFT	LH9124	1.7	26.2
6.0 RCMC Shift and Interpolation	LH9124	4.4	10.0
7.0 Azimuth multiply, 8.0 Azimuth IFFT	LH9124	2.2	6.3
9.0 Azimuth Resampling	LH9124	5.3	8.4
10.0 Range Resampling	LH9124	7.4	-
11.0 Detection	GP CPU	-	-
12.0 Calculate Range Filter and SRC	GP CPU	-	-
13.0 Calculate Doppler Centroid	GP CPU	-	-
14.0 Calculate RCM Shifts and Indices	GP CPU	-	-
15.0 Calculate Ambiguity	GP CPU	-	-
16.2 Azimuth filter FFT	LH9124	1.7	-
16.1, 16.3, 16.4 Calculate Azimuth Filter	GP CPU	-	-

As can be seen from the table, the resampling operations take up 45% of the total execution time. Without the resampling operations, the architecture is able to achieve 17% of real-time. With the resampling, only 9.3% is obtained. To achieve 1/10 real-time operation with resampling, two LH9124s in a parallel configuration are needed.

The data transfer rates between main memory and the buffer at the LH9124 input port are shown in the right-most column of Table 9.2. The highest data transfer rate between main memory and the buffer occurs for the azimuth FFT stage of the algorithm. This is due to the fact that the azimuth FFT stage performs the least amount of computation between data transfers. The maximum transfer rate is 26 Mwords/s or 150 MB/s. This is higher than that achievable with normal DRAMs. However, a memory that makes use of a combination of wide data paths, interleaving, and fast RAMs can reach this transfer rate.

The computation requirements on the general purpose CPU are 36.3 MOP/s if it is to keep up with the LH9124 on the range-Doppler algorithm with resampling. This is close to the performance available on a single chip DSP. However, it is doubtful if this level of performance could be sustained on a single CPU if it is also responsible for higher level control functions.

9.3.4 Multi-LH9124 Processor Architectures

The LH9124's data path oriented architecture makes it fairly well suited to parallel configurations. The many possible configurations can be grouped into two general categories.

Fine grain

Fine grain approaches use multiple LH9124s to speed up fundamental operations. Cascade or parallel configurations are used with interconnections that support pipelined operations and parallel FFTs. The advantage of this approach is higher throughput and near-linear speed-up. The main disadvantage is the potential lack of flexibility. For example, the speed-up of cascaded architectures depends on the FFT length. Thus, good speed-up may not be available for all algorithms.

Coarse grain

Coarse grain approaches use replicated LH9124 based processor-memory subsystems. These PEs run in parallel on distinct data subswaths. This type of architecture is much more general, however, it may not be as efficient as a fine grain approach.

A combination of coarse and fine grain approaches may also be used. Some possible interconnections for both coarse and fine grain parallelism are shown in simplified form in Figure 9.9. Each PE in the figure represents an LH9124 and its associated buffers and address generators.

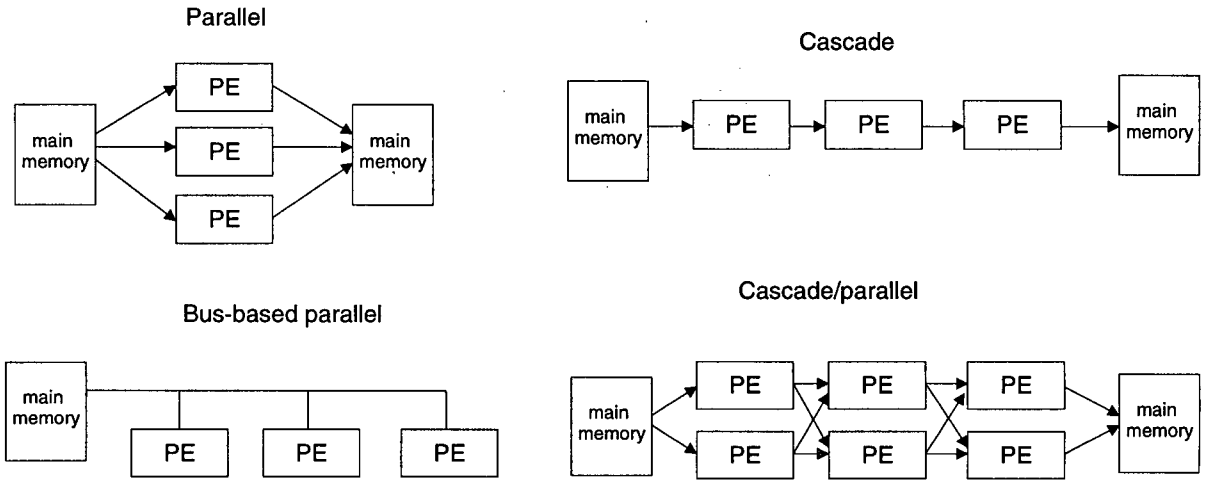


Figure 9.9: Some options for LH9124 parallelism

It should be noted that if the number of LH9124s is increased, the number of general purpose CPUs in the system will have to increase by a similar amount.

9.3.5 Range Doppler Algorithm on Multi-LH9124 Architectures

This section examines the parallel execution of the range-Doppler algorithm described in Section 2.5.4 on an architecture composed of multiple LH9124 processors. Each PE is assumed to be an LH9124 subsystem like the one shown in Figure 9.7. It is again assumed that the memory and interconnect can keep up with the PEs.

The first method of parallelism that is examined is the coarse grain horizontal partitioning modelled in Section 2.6.4. The same allocation of tasks to the LH9124 as in the single processor model is used. The time taken by each processor is computed using the vector sizes from Section 2.5.4. Figure 9.10 shows a graph of the percentage of real-time achieved for varying numbers of LH9124s. This graph is similar to the ones obtained in Section 2.6.4, with the same discontinuities caused by the jumps in range FFT length. With resampling, real-time operation requires 15 LH9124s. Without resampling, real-time operation is attained with 10 LH9124s.

As before, the main problem with this approach is the inefficiency caused by the data overlap due to the relatively long range matched filter. Some possible solutions to this problem were mentioned in Section 2.6.4. If each PE were faster, then larger range blocks could be used. The PEs can be made faster by either using faster processors, like an LH9124 with a higher clock speed, or by using multiple LH9124s for each range subswath.

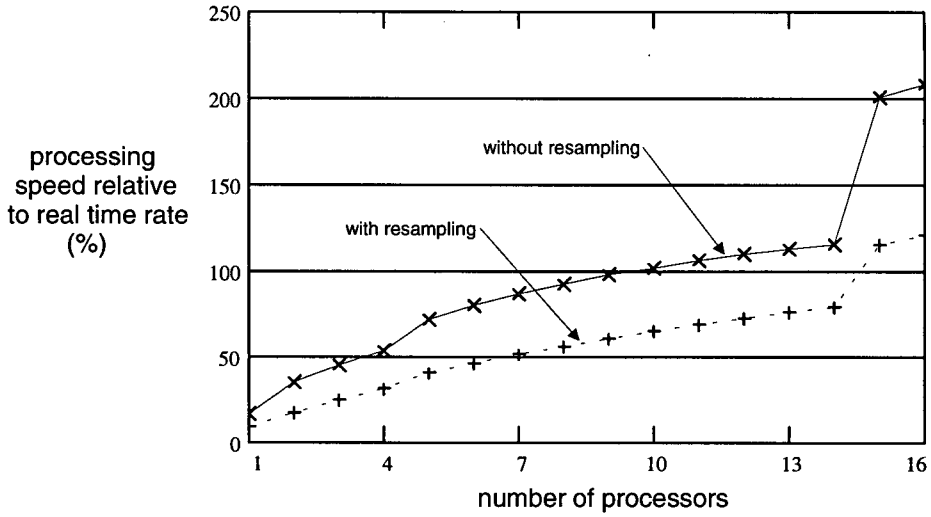


Figure 9.10: Performance achieved for various numbers of LH9124 processors on the range-Doppler algorithm

A method of partitioning that might improve the efficiency of the parallel architecture is to use dynamic horizontal partitioning. This method may require the use of a centralized corner turn memory. This would imply a system architecture like the one shown in the top half of Figure 9.11. Each PE is an LH9124 sub-system. This architecture was modelled and the results are shown in Figure 9.12.

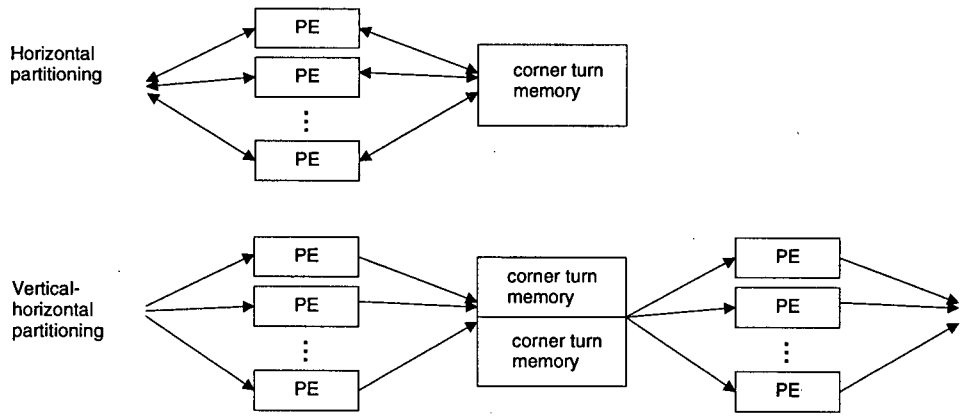


Figure 9.11: Simple architectures with corner turn memories

Figure 9.12 shows that 11 LH9124s are required for real-time operation with resampling. Fewer processors are required than for the no corner turn approach. The penalty here is the cost of the corner turn memory. It has been assumed that the corner turn memory can supply data fast enough to keep up with the LH9124s. This may be hard to achieve in practice.

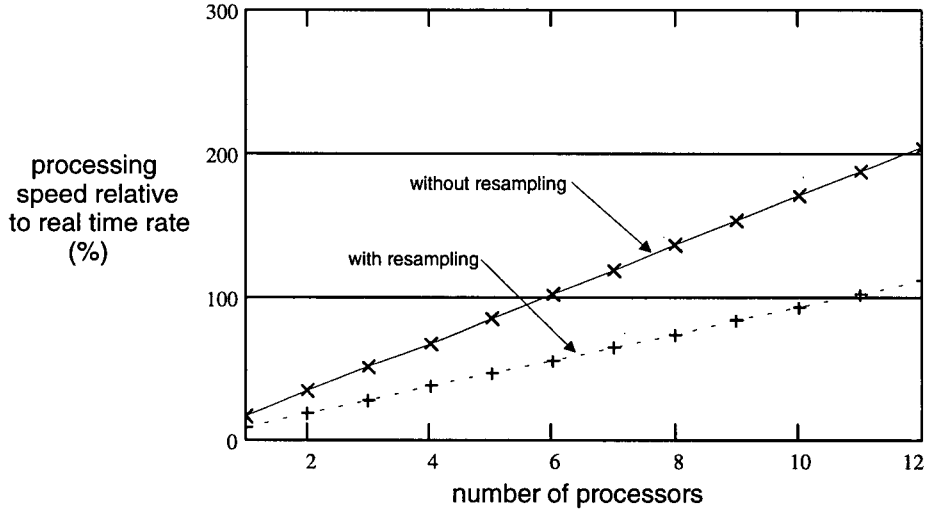


Figure 9.12: Performance of horizontal partitioning with a corner turn on multiple LH9124 processors with the range-Doppler algorithm

Another partitioning method that can be used is vertical-horizontal partitioning. This scheme also requires a centralized corner turn memory and allows range compression and azimuth compression to be pipelined. This implies that the corner turn memory be double buffered or dual-ported as is shown in the bottom half of Figure 9.11. This architecture was modelled and the results are shown in Figure 9.13. The percentage of real-time operation achieved is plotted against the number of processors in each pipeline stage. Separate graphs are shown for range and azimuth compression. Azimuth compression is taken to mean all the operations after the corner turn in range-Doppler.

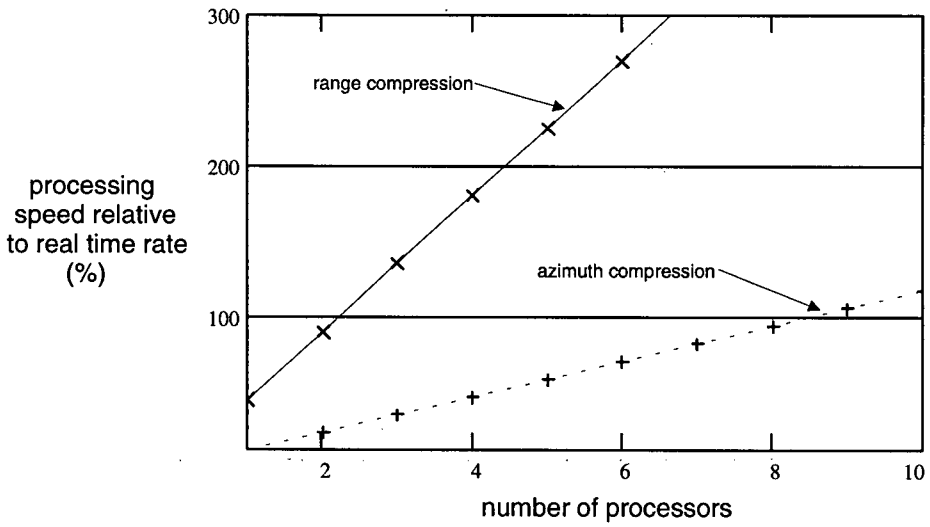


Figure 9.13: Performance of vertical-horizontal partitioning with a corner turn on multiple LH9124 processors with the range-Doppler algorithm

Figure 9.13 shows that for real-time operation, three LH9124s are required for range compression and nine for azimuth compression for a total of 12. The central corner turn memory is again a potential bottleneck in the architecture. At real-time operation, the transfer rate into the corner turn memory is about 48 MB/s or about 8 MW/s. This data rate probably cannot be sustained with a commercial memory but can be relatively easily accomplished in a custom design.

It should be emphasized that the models used in this section are quite rudimentary. The purpose of the performance estimates is only to give an idea of the feasibility of building a real-time SAR processor using an LH9124 based approach. The conclusions reached are quite positive. With about 11-15 LH9124s and a similar number of general purpose CPUs, a real-time SAR processor for the range-Doppler algorithm for Radarsat can be constructed. This could conceivably fit in one 19" card cage. This is no small feat considering the sustained computation is on the order of 4.5 GOP/s.

9.3.6 Other Algorithms

The LH9124 DSP was also applied to models of some other SAR processing scenarios. The first is the algorithm used in the MDA IRIS X2C airborne SAR processor. A mapping of algorithm steps to either the LH9124 or a general purpose CPU was performed in a similar manner to the one shown in Table 9.2. A single LH9124 configuration is able to process a 4k x 2k frame in approximately 6.4 seconds. Since the frame time is 4.5 seconds, at least two LH9124s are needed to meet real-time operation. The operation rate required of the GP CPU is 7.3 MOP/s which is within the capabilities of a single chip CPU. Thus, a possible system would consist of 2 LH9124s and at least 1 GP CPU (most likely about three or four) on a total of about 4 or 5 boards. This should be compared with the actual system that was built for this application using VASP boards based on the DSP96000 general purpose DSP. The system consisted of 18 General Signal Processor boards with 4 processors each, four I/O Processor boards with one processor each, and four Two-Port Memory boards. This gives a total of 76 processors and 26 boards.

The lack of interpolations makes the chirp scaling algorithm particularly well suited for implementation on the LH9124. Chirp scaling requires more vector multiplies, but these can usually be performed with no overhead when they are rolled into the first stage of an FFT. A chirp scaling implementation on a single LH9124 that uses the same parameters as the range-Doppler model of Section 2.5.4 can achieve almost 25% of real-time operation. This is without any post-processing resampling operations. Chirp scaling requires similar data transfer rates as range-Doppler. It also suffers from the fact that extremely high memory to buffer data transfer rates are necessary for algorithm steps that involve small amounts of processing per vector. In chirp scaling, like in range-Doppler, this occurs for the azimuth FFT.

9.3.7 Vector DSP Conclusions

The use of the LH9124 DSP has been shown to be a promising implementation alternative for SAR processing. This chip provides a fairly straight-forward solution for high performance vector processing. Despite the high performance attained by the chip, it is not fast enough for real-time processing and parallel architectures are needed. The design of the memory system and control structure are left open by the architecture of the LH9124 and will be major challenges in the design of an LH9124 based processor. A practical design that incorporates the LH9124 is examined in Section 9.7.

Some of the shortcomings of the LH9124 are discussed below.

- Numerous additional chips are needed to support the LH9124. External RAMs and address generators are required at each port. If these functions were incorporated on the main chip the system design would be considerable simpler. Butterfly DSP is attempting to address this by building MCMs that combine the LH9124 with some support chips.
- All four data ports are rarely necessary in a system due to the cost of all the support circuitry for each port. For most applications, two ports would be enough: one for data and one for coefficients.
- The FIR filtering speed of the LH9124 is disappointing. For a fixed algorithm like FIRs, much higher performance should be possible. The LH9124 is no faster per filter tap than general purpose DSPs which can also usually handle one filter tap per cycle. The only performance advantage of the LH9124 is that it can handle complex MACs in a single cycle. A possible work-around for this problem is to use the LH9124 in conjunction with a fast digital filter chip.
- The need for separate scalar CPUs complicates the design.
- More flexible address generation circuitry is needed. The LH9230 cannot handle the more complex addressing patterns that are required for sample rate conversion. Also the LH9230 cannot handle out-of-the ordinary situations. For example, it would be convenient to be able to perform filtering across vectors stored in the buffers since this would speed up RCMC in range-Doppler. Currently the only way around this problem is to design a custom address generator. A more flexible solution would be to make the address generator programmable like a microsequencer. This way the programmer could choose any addressing sequence.

The LH9124 is not the only chip in its class. In fact it seems that a number of new chips optimized for fast FFTs will become available shortly. Texas Memory Systems has announced the TM66 swiFFT. It has a slightly faster 1k CFFT time than the LH9124 but uses full 32-bit IEEE floating point arithmetic. The TM66 architecture is quite different than the LH9124. It also makes extensive use of pipelining, but the

data flows in only one direction, from two input ports to one output port. Butterfly DSP, who have taken over the support of the LH9124 from Sharp, claim to be working on the BDSP32000, which will feature a 32-bit fixed or floating point 1k CFFT time of 10 μs.

9.4 Optimized DSP Architecture

Based on the experience gained from studying SAR algorithms and existing DSPs, it is possible to characterise a DSP IC that has an architecture optimized to handle the requirements of SAR processing. This section examines some aspects of optimized DSP architectures.

A natural question that arises is whether or not it is possible to build a single chip processor that can handle all the arithmetic needed for SAR. The requirements on such a processor can be taken from the range-Doppler model of Section 2.5.4. The amount of computation in the main processing flow (steps 2.0 to 8.0 in Figure 2.17) is on the order of 3 GOP/s. The data transfer rates are upwards of 50 MW/s. We are quite close to realizing this level of performance. One of the fastest FFT processors today is the LH9124 which achieves an execution rate of 15 OPs/cycle when executing FFTs. One of the fastest general purpose processors available today is the DEC Alpha 21064 which runs at clock speeds in excess of 200 MHz. If these features were to be combined, one would obtain a computation rate of 3 GOP/s which is what is required for a single chip real-time SAR processor. Of course, the LH9124 only achieves this high operation rate for the fixed FFT algorithm. But the intent here is to show that the technology exists today for a single chip real-time SAR processor. A possible supporting example is the new chip that Butterfly DSP is claiming to be developing that can perform a 1k CFFT in 10 μs. This translates into an operation rate of 5.1 GOP/s.

The attributes of an ideal SAR processor chip can be based on solutions to the shortcomings of the LH9124 discussed in the previous section. The features listed below would be desirable in a SAR processing chip.

- A high clock speed
- A vector processor that supports extremely fast FFTs and other vector operations: Like the LH9124, this would be achieved using multiple arithmetic units and extensive pipelining. The vector processor should support complex IEEE floating point numbers.
- SRAM double buffers large enough to hold the longest data vector expected
- Programmable address generators for the inputs and output of the vector processor
- Efficient support for filtering: It should be possible to process at least eight or more complex filter taps per clock cycle

- A scalar processor: This could be similar to one of the existing general purpose floating point DSP chips. This processor would also be responsible for the control of the vector processor, eliminating the need for a separate scheduler.
- Sufficient SRAM to minimize off-chip access for both program and data by the scalar processor

It is not possible to achieve all this with today's technology. However, it is not unrealistic for the near future, especially considering TI's achievement in putting five DSP CPUs on one chip in the C80 MVP.

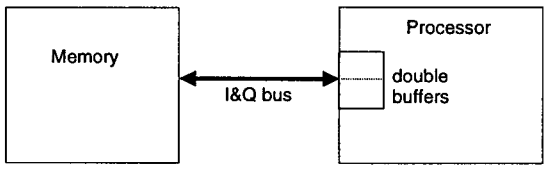


Figure 9.14: Hypothetical single chip SAR processor

As the speed of a single chip processor is increased, the requirements on the data path elements that route data to and from the processor also increase. A simple estimate of the overall transfer rate for a real-time SAR processor can be obtained by making the following assumptions: (see Figure 9.14).

- One corner turn operation is required.
- The complete data array is transferred between the external memory and the processor four times (once at the beginning, twice for the corner turn, and once at the end).
- Parallel transfers of complex words are possible. (This requires a 64-bit wide data path for 32-bit data.)
- Data transfers are fully overlapped with computation; i.e. the full processing time is available for data transfers.
- The data volumes from Section 2.5.4 are used.

These assumptions result in a data transfer rate of 31 MCW/s. This fairly low number results because we are assuming very few data transfers are necessary and that a large amount of computation happens between data transfers. This is attainable with existing DRAMs and an appropriate interleaving scheme. Much larger data transfer rates are needed between the vector buffers and the processing unit. This indicates that these buffers should reside on the same IC as the processing unit.

Combining so much functionality makes for a very large IC. It may be more cost effective to use an alternate approach and use fine grained parallelism to spread out the functionality over multiple chips. This requires extremely fast inter-chip communication. No currently available chip supports this. This option is discussed further in Section 9.6.

9.5 Digital Filter Chip Architecture

The appearance of extremely high performance digital filter chips has made the time domain approach to SAR processing more appealing. The highest performance digital filter chip encountered to date is the Zoran ZR33288 Video Rate Digital Filter. Some of its key characteristics are listed below.

- 8 bit data input
- 288 coefficients (10 bit)
- 14.3 MHz sample rate for all filter lengths
- real data only (Four chips plus adders are needed to handle complex data.)
- Longer filter lengths can be handled by cascading a number of chips.

The Zoran chip is capable of about 8 GOP/s which is the highest performance of any currently available chip surveyed. This makes it about 13 times as powerful as the fastest FFT chip which has a peak performance of about 600 MOP/s (but with a larger word length). A potential application of the Zoran chip to a quick-look SAR processing situation is described in [68]. A board is described that makes use of two sets of four Zoran chips in a pipeline configuration. The design is able to achieve 2/3 real-time operation for high precision Radarsat processing (without quadratic RCMC) on 2048-sample range lines.

This section examines the ZR33288 based time domain approach to SAR processing. It is based on [68] but adds a performance analysis, some design additions, and an overall analysis.

9.5.1 Architecture

A block diagram of a filter-chip based SAR processor is shown in Figure 9.15. The input FIFO acts as a buffer for the incoming data and serves to equalize the input data rate with the data rate supported by the digital filters. The high level design of a complex filter block is shown in Figure 9.16. The same block is used for both range and azimuth processing. It is made up of four ZR33288 digital filter ICs. Longer filters than the 288 taps supported by the filter chips can be accommodated by feeding the outputs back to the inputs through FIFOs that provide the appropriate amount of delay. Filter coefficients are stored in a coef-

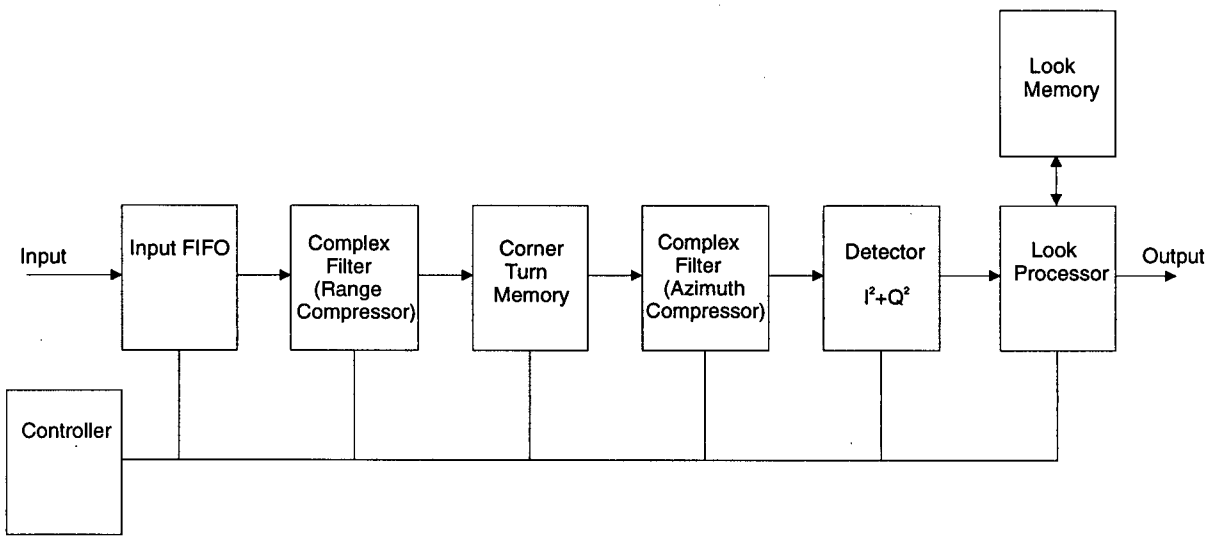


Figure 9.15: Time domain SAR processor block diagram

ficient memory and are written to the filter chips by the control logic. The coefficient memory can be quite large since different filter coefficients are potentially needed for each vector.

The corner turn memory can be either single or double buffered using the methods described in Section 7.3.2. Low resolution processing may call for each range line to be subsampled before it is stored in memory. The size of the memory can be chosen to match the size of the image being processed. The data rate in and out of the memory is 14.3 MHz, which is equivalent to a cycle time of about 70 ns. Since the data needs to accessed at this speed in both row and column directions, existing DRAMs are too slow and SRAMs must be used.

The look processor stores the values from the initial look in the look memory. Each subsequent value is added to the one in the look memory and the new value is stored in the look memory. Once all the looks have been summed, the data is read out of the look memory in the range direction, and a deskew filter is applied to remove the effects of the shifts applied during range processing.

A useful refinement to the basic time domain processor design is the addition of buffers and more sophisticated address generation logic at the inputs and outputs of the complex filters. This would facilitate the implementation of sample-rate conversion. This is required if the image needs to be resampled. In the time domain approach, this can be combined with range and azimuth processing. A possible structure is shown in Figure 9.17.

Such a filter based resampling structure might also be a useful addition to a fast convolution SAR processor. It could be used to off-load the resampling operations from the main processors. This may be worth-

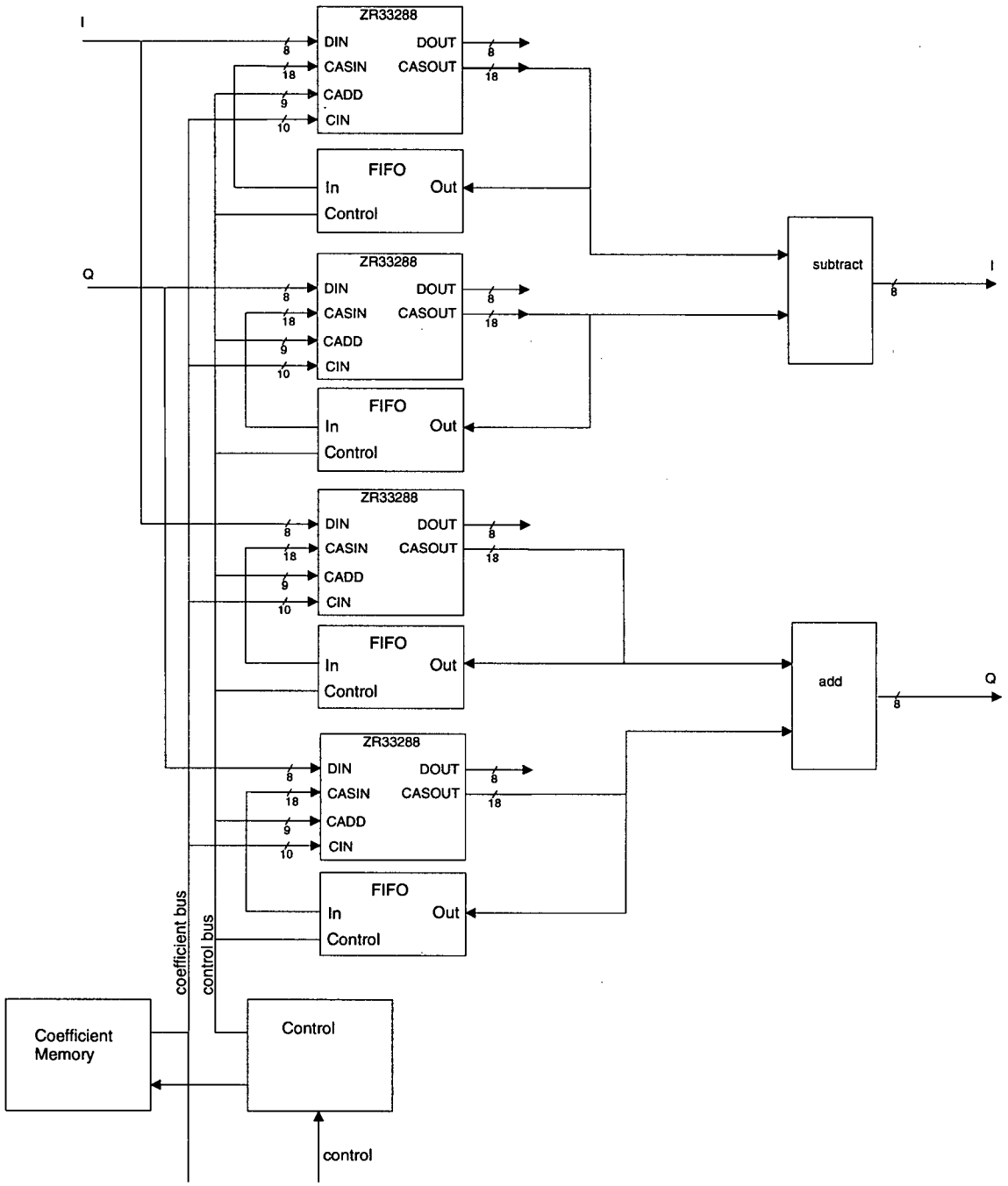


Figure 9.16: Time domain complex filter block

while since resampling takes up a large portion of the total number of operations and since fast FFT chips are usually not as efficient for filtering.

9.5.2 Range and Azimuth Processing

Range compression is performed using standard time domain convolution. Matched filter lengths of up to 288 taps can be accommodated in a single pass through the ZR33288. Longer filter lengths require multi-

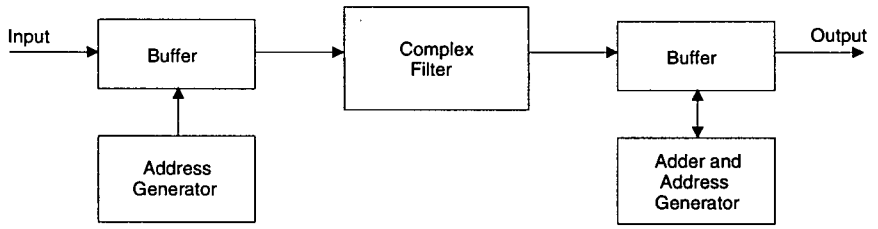


Figure 9.17: Modification to complex filter for sample-rate conversion

ple passes. Linear RCMC is accommodated by using different filter coefficients for each range line. The coefficients implement a shift that removes range walk. This shift must be removed after azimuth processing.

Azimuth compression is handled similarly to range compression. Quadratic RCMC is handled by processing multiple input lines for each output line using appropriate coefficients that extract the energy that belongs in the output line from each of the input lines. Multi-look processing is handled by processing each azimuth line once per look.

9.5.3 Performance

The performance of the time domain processor described above was modelled and the results are shown in Table 9.3.

Table 9.3: Time domain processor performance

Scenario	% of RT for RC	% of RT for AC
high resolution processing		
1 look	55%	59%
4 looks	55%	15%
4 looks with quadratic RCMC	55%	2.5%
low resolution processing		
1 look	180%	1000%
4 looks	180%	250%

The SAR scenario from Table 2.12 is used for high resolution processing. Quadratic RCMC is assumed to require the processing of six adjoining azimuth lines. For the low resolution processing scenario the parameters from Table 2.12 are used again with the following exceptions: L_r=100, L_a=60, N_{az}=1024, and the output range line length is 1024.

The throughput of the time domain processor falls off rapidly for higher resolution processing. However, the processor is well suited to a low resolution four look processing situation. This situation is also suited for an implementation using the SPECAN algorithm. In order to process the same data using SPECAN, about 30 MOP/s of computation are required. This could be achieved using two or three general purpose DSPs. Some advantages of both approaches are given below.

- general purpose DSPs
  - much greater flexibility
  - much lower hardware development cost
  - greater precision due to longer word length
- digital filter processor
  - more predictable performance and scalability
  - able to use longer filter lengths and still achieve real-time speed
  - possibility of lower repeat costs

9.5.4 Analysis

Some of the main advantages of the filter-chip based time domain approach to SAR processing are listed below.

- Digital filter ICs offer the highest operation rate per chip available today.
- Scalable architectures can be built with predictable performance.
- A range of processing situations can potentially be accommodated: from low to high resolution.
- Output image resampling can be handled more efficiently than in a frequency domain processor.

Some potential disadvantages surrounding filter-chip based SAR processors are summarized below.

- A great deal of custom hardware design is required.
- The amount of custom hardware makes it likely that a time domain design intended for high performance will exhibit low flexibility.
- Many chips only handle short word lengths. This results in either lower precision processing or a higher chip count. The design is also complicated by the need to carefully track how the data is scaled.
- Most filter chips do not directly handle complex values. This results in a higher chip count.
- A design is likely to be very specific to a particular digital filter device.

- The control requirements for the filtering structures can be relatively complex.
- Control and coefficient generation needs to be handled by a separate CPU.
- In a pipelined design, it can be difficult to balance the amount of processing between the range and azimuth stages.
- The efficiency of the time domain approach can decrease rapidly for higher precision processing due to longer filter lengths, and due to the extra processing that is required once effects like RCMC need to be incorporated.
- Multi-look processing requires additional processing, and as a result is harder to accommodate than in a frequency domain approach.

The viability of the time domain approach is dependent on the matched filter length. If the matched filter is short and the vector length is relatively long, then a time domain approach can be efficient relative to a frequency domain approach. This applies primarily to low resolution processing where filter lengths are shorter. For example, with a filter length of 100 and vector length of 1024 the time domain approach only requires about eight times as much computation as fast convolution. However, for a filter length of 600 and a vector length of 8k, the time domain approach requires about 35 times as much computation.

9.5.5 Digital Filter Conclusions

The digital-filter chip based time domain SAR processor has been shown to be most suitable to high speed low resolution (greater than about 50 m) processing. Specifically, the time domain processor is suited for applications that require:

- relatively short matched filter lengths (less than 288 for the ZR33288)
- no long wordlengths
- a small number of looks
- no quadratic RCMC

For a processing problem that has these characteristics, the time domain processor can require significantly less hardware than an equivalent fast convolution based processor. For example, for medium resolution single look processing with no quadratic RCMC, two time domain processors in parallel would be able to achieve real-time operation for Radarsat. This might otherwise require upwards of 40 general purpose CPUs or six vector DSPs like the LH9124.

Performance falls off rapidly with longer filter lengths, larger numbers of looks and quadratic RCMC correction. As a result, in most high resolution cases there is no real performance advantage to using the time domain approach. Any speed advantage is outweighed by the lack of flexibility. The advantage presented by the high speed of the digital filter chips is also partially offset by the availability of efficient frequency domain processing algorithms, especially for low resolution processing.

The dedicated filter chip approach to SAR processing would be significantly more attractive if some of the flexibility and fixed wordlengths problems could be overcome. This may be addressed through the use of FPGAs to build custom hardware filter devices. This option is addressed in Section 9.8.

9.6 Fine Grained Parallel Architecture

Fine grained architectures have a number of advantages over the coarse grain architectures considered so far (as previously mentioned in Section 2.6.2). Fine grained DSP architectures

- have more opportunities for parallelism,
- are easier to program/schedule, and
- are suitable for automatic programming tools.

In order to build a fine grained SAR processor, the appropriate building blocks must be made available. Current microprocessors and DSPs do not have the level of support for high speed, low latency communication that is necessary for fine grained processing. Chips that address this issue have been proposed but have not been produced commercially. An example is the SIMC (Special Instruction Set Multiple Chip Computer) proposed in [62], which is based on the TMS320C40, but which incorporates features that make it more suitable for fine grained parallelism.

Fine grain approaches are useful with existing processors in the following situations:

- It is not possible to use coarse grain partitioning to achieve the required performance level.
- The partitioning allows all the data required for an operation to fit into on-chip memory. This can result in super linear speed ups. Examples of this for the case of splitting the 1D FFT over multiple C40s is given in [85].

Due to the lack of building block components, no fine grained system level products are available. As a result of this lack of commercial products, not much experience exists in implementing large scale signal processing applications on fine grained machines.

Fine grained signal processors make more intensive use of software tools. A typical programming design flow might entail the following four steps, the last three of which can be automated.

- model the algorithm in terms of a signal flow graph
- obtain a schedule
- find a processor mapping
- translate into machine code

One area that will see the emergence of massively parallel fine grained architectures is the area of FPGA computing covered in Section 9.8. FPGA architectures may be one of the main beneficiaries of the research in fine grained parallelism.

The potential advantages of fine grain architectures are mitigated somewhat by the fact that the large data set sizes used in SAR processing make coarse grain partitioning relatively easy. Fine grain approaches may be more attractive for extremely high performance applications where coarse grain architectures become less efficient.

In summary, fine grained architectures are not found to be a viable alternative for most SAR processing applications for the following reasons.

- **Availability:** Both the hardware and software components necessary for a fine grained design are not available commercially.
- **Cost:** The implementation of a SAR processor on a fine grained processor would entail the design of a completely custom processor.
- **Risk:** Fine grain processors are not a very mature technology.

9.7 A Heterogeneous Architecture

This section describes the architecture and analysis of a proposed SAR processor design that combines some of the design approaches described in the previous sections. The design makes use of both vector and scalar DSP chips and is referred to as a vector/scalar architecture (VSA). The goal of the design is to achieve 1/10 real-time processing using the scenario of Section 2.5.4 on images up to 8k × 4k in size.

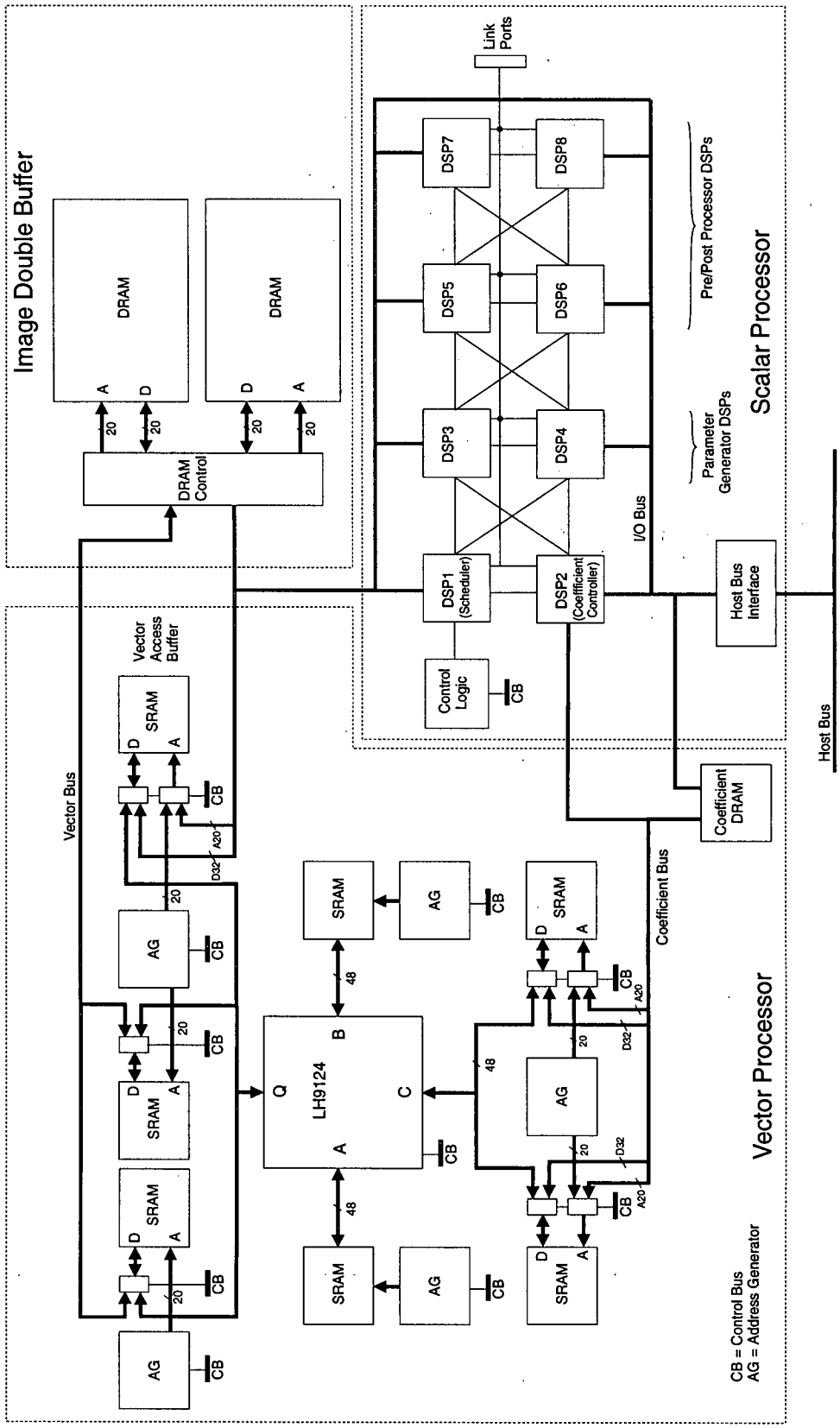


Figure 9.18: Block diagram of the vector/scalar architecture

A block diagram of the architecture is shown in Figure 9.18. The design can be divided into three parts:

- the vector processor,
- the scalar processor, and
- the image double buffer.

Some of the key architectural features of the VSA are given below:

- Vector processing is separated from scalar processing.
- Vector and scalar processor data flows are decoupled via twin DRAM banks.
- High bandwidth data flows are assigned their own buses.
- The computational elements and the data paths are flexible enough to handle a range of processing situations.
- The performance goals are achieved while using commodity DRAMs for all large memories.

9.7.1 Vector Processor

An LH9124 based design is used as the vector processor for the system. The double buffers attached to the data ports of the LH9124 are at least 48 kB for 24-bit data or 64 kB for 32-bit data for each half of the double buffer. An additional buffer is attached to the data port and is accessible from the I/O bus. This buffer allows the scalar DSPs to directly supply data to the vector processor. This vector access buffer is made somewhat larger to give the general purpose DSPs more time to access the data. A possible size is $128\text{k} \times 48$. At least 15 ns SRAMs are needed for a 40 MHz LH9124.

The buffer attached to the coefficient port of the LH9124 is organized as multiple pages. The purpose of this is to reduce the otherwise very large bandwidth requirements on the coefficient bus. When a new set of coefficients is needed, a switch is simply made between pages. If 16 8k coefficient vectors are stored, the size of the buffer is $128\text{k} \times 48$ or 768 kB.

The coefficient DRAM provides storage for coefficient vectors. If all the azimuth matched filters are prestored in the memory and a new filter is used for every four azimuth lines, the memory will need to be at least 48 MB in size. Fast page mode access can always be used for this memory since all accesses are made to consecutive vector addresses and no corner-turns are required. This memory is accessible from both the coefficient bus and the I/O bus. The intent is to allow either the coefficient controller DSP or the host computer to update the coefficients.

9.7.2 Scalar Processor

The scalar processing devices are general purpose DSPs like the C40 or the SHARC. The design relies on their level of performance and inter-processor communication capabilities. In the remainder of this description, the scalar processor is taken to be a mesh of SHARC chips. The large on-chip memory of the SHARC is particularly attractive in this design. Since the entire data array never has to be distributed among the DSPs, the on-chip memory is large enough to serve as a buffer and no external memory is necessary. If the C40 is used instead, each PE would require memory external to the processor. However, the C40's dual bus structure would greatly simplify the interfacing between buses in the system.

Functionally, the scalar processor can be divided into two portions: fixed functions and non-fixed functions. The two fixed functions of scheduler and coefficient controller are fixed by the hardware design to two distinct DSPs. The non-fixed functions, like pre/post-processing and parameter generation, can be performed by any combination of the remaining DSPs. The scalar processor is very scalable since the number of general purpose DSPs can be changed relatively easily.

9.7.3 Image Double Buffer

The image double buffer is composed of two identical banks of DRAM organized as 32 bit words. Each bank can be up to 256 MB in size. The most efficient way to implement this memory is to use 32 MB SIMMs. Eight SIMMs are required for each bank. Accesses to the image double buffer from the I/O bus are always to consecutive addresses allowing the fast page mode cycle time to be used. Accesses from the vector bus can be to either rows or columns of the memory. As a result, the slower random access cycle time is assumed for all accesses from this bus.

The memory controller is responsible for managing the image double buffer. It implements a crossbar switch that allows independent access of either memory bank from either bus. It is also handles

- DRAM access timing, parity (or ECC), and refresh,
- the vector bus interface including address generation, and
- the I/O bus interface.

It is not necessary to use interleaving to meet the performance goals. However, the design of the memory controller should be able to accommodate the future addition of interleaving which may be necessitated by an increase in clock rate in either the vector or scalar processor. The memory controller would most likely be implemented in a PLD.

9.7.4 Data Movement

The VSA is designed to allow high speed data transfer with no bottlenecks. To achieve this, there are four main buses in the VSA: the I/O bus, the vector bus, the coefficient bus, and the control bus.

The I/O bus is a general purpose bus that is used for many different data transfers in the system. None of these are high bandwidth transfers. In order to keep the latency for all the different users of this bus low, the utilization of the bus is kept low. The I/O bus has a high bandwidth; it runs at a speed of about 30 MHz. The clock speed is limited by the fast page mode cycle time of the DRAMs in the image double buffer. The I/O bus is also connected to the host bus interface, which can be a standard PCI or VME interface chip.

The vector bus connects the LH9124 to the image double buffer. It carries the high bandwidth traffic that fills and empties the LH9124 buffers. The vector bus utilization is high, but since it is point-to-point and its workload can be completely predicted, this is not a problem. The available bandwidth of the vector bus is limited by the need to accommodate random access to the image double buffer. As a result, its average transfer rate is about 10 MHz.

The coefficient bus services the LH9124's coefficient buffer. The traffic is similar to that on the vector bus.

The control bus is a special purpose bus that carries the control signals for the various chips in the design. It originates in control logic that is attached to the scheduler DSP.

9.7.5 Implementation

The VSA should fit on 1.5 to 2 full-length PCI bus or 6U VME bus boards depending on how aggressively it is packaged. The use of daughtercards may make it possible to put everything in one board slot.

It is also possible to assemble an architecture similar to the VSA using mainly commercial boards. The following products are needed:

- an LH9124 board (like the PC/M VSP-9 or the Catalina CRV1M40)
- a memory board to support the LH9124 board
- a general purpose DSP board with multiple SHARCs or C40s (available from a number of vendors)

The functionality of the image double buffer would be more difficult to find in a standard product.

A VSA design based on COTS products would have to be carefully evaluated since it is hard to achieve optimal performance from a mix of standard products. It would be especially difficult to match the high data transfer rates achieved by the VSA's multiple bus design using commercial products.

9.7.6 Overall System

The VSA can form a part of an overall SAR processing system like the one depicted in Figure 8.22.

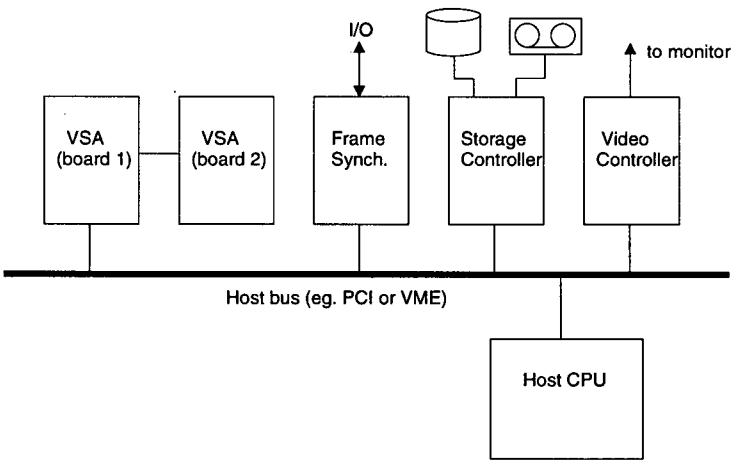


Figure 9.19: VSA based SAR processing system

A high bandwidth host bus is required as the backbone for the system. Either PCI or VME64 would be good choices for this bus since they offer sufficient bandwidth and the opportunity to use standard PC or VME platforms. It is assumed that the SAR data is transferred over the host bus before and after it is processed. While this places significant loads on the host bus, it allows standard mass storage interfaces to be used. No other time critical SAR processing related traffic is carried by the host bus.

The following subunits are connected to the host bus:

- VSA board(s)
- frame synchronizer (if necessary)
- high speed mass storage interface (to disk and tape)
- video controller to interface to a high resolution monitor (not shown): if necessary
- host CPU interface (additional peripheral devices can be attached to the host CPU via a lower speed peripheral bus like ISA)

9.7.7 Range-Doppler Algorithm

This section describes the mapping of the range-Doppler algorithm of Section 2.5.4 to the VSA. The amounts of computation and data set sizes are taken from the analysis of the algorithm in that section.

Scalar Processor

The raw SAR data is transferred over the host bus to general purpose DSPs which handle:

- input data unpacking
- I/Q balance (step 1.0 of Figure 2.17)
- number format conversion
- data transfer into the image memory buffer

When both the scalar processor and the vector processor have completed processing the data in one of the image memory banks, the banks are switched. The scalar processor general purpose DSPs then perform:

- format conversion: block floating point to floating point
- azimuth resampling (step 9.1 of Figure 2.17)
- range resampling (step 10.1 of Figure 2.17)
- detection (step 11.0 of Figure 2.17)
- radiometric correction
- output format conversion
- data transfer out of the VSA over the host bus

Once all this is complete, the general purpose DSPs ingest a new frame of data and the process begins again.

These operations are performed by the four general purpose DSPs that make up the pre/post-processor portion of the scalar processor. When predicting the performance of the general purpose DSPs, an operation rate of 30 MOP/s is assumed. This is 25% of the claimed peak rate of 120 MOP/s and about 38% of the claimed sustained performance figure of 80%. This a fairly safe assumption since these PEs are performing fairly fixed algorithms consisting of filtering and arithmetic functions with little variation. When calculating data transfer times, a transfer rate of 30 MB/s is assumed for the I/O bus. It is assumed that there is no

overlap between computation and data transfer. With these assumptions, the total time to perform all the functions listed is about 21 seconds, which is about 12% of real-time.

The coefficient and filter generation functions are performed by the two parameter generator DSPs in the scalar processor. These DSPs perform steps 9.2, 10.2, 12.0, 13.0, 14.0, 15.0, 16.1, 16.3, and 16.4 of Figure 2.17. The azimuth matched filter is assumed to be updated every four lines. The computation rate required is 19 MOP/s for 1/10 real-time. This is easily achievable on two DSPs.

Vector Processor

The timings for the functions performed by the vector processor are shown in Table 9.4.

Table 9.4: Times for vector processor operations

Step in Algorithm	Time required (sec)
2.0 Range FFT, 3.0 Range multiply, 4.0 Range IFFT	
Processing	5.8
Data transfer	5.7
5.0 Azimuth FFT	
Processing	1.7
Data transfer	4.4
6.0 RCMC	
Processing	4.4
Data transfer	4.4
7.0 Azimuth multiply, 8.0 Azimuth IFFT	
Processing	2.2
Data transfer	4.4
16.2 FFT of azimuth matched filter	
Processing	0.4
Data transfer	0.1
Total:	19.5
Time allowed for 1/10 RT:	26 sec.

The data is transferred between the image double buffer and the LH9124 buffers for each step in the table. With the exception of step 16.2, the data transfer times are based on a data transfer rate of 10 MHz from the image double buffer. The data transfer for step 16.2 is assumed to occur between the general purpose DSPs and the vector access buffer and so it occurs at a rate of 40 MHz. For each step, the larger of the computation time and the data transfer time is taken as the total time.

The data transfer and computation times are well balanced for all steps except for the azimuth FFT and IFFT. This imbalance results because the data is transferred back to memory after the azimuth FFT so that it can be read out in the range direction for RCMC. The delay introduced by these data transfers means that the LH9124 is not fully utilized. Fast access modes of the DRAMs cannot be used to reduce this delay unless an explicit corner turn is performed. One way this delay could be reduced is by introducing interleaving in the image double buffer at the expense of a more complex memory controller.

9.7.8 Vector/Scalar Conclusions

The analysis of the VSA has shown that it is capable of handling 1/10 real-time satellite SAR processing using the range-Doppler algorithm of Section 2.5.4. A similar analysis was carried out for the IRIS X2C airborne processing situation. A single VSA should able to handle the all IRIS X2C processing requirements.

It is interesting to compare the VSA with a general purpose DSP only design. The LH9124 could be replaced by about 8 to 10 general purpose DSPs. This would result in a more general, flexible, and standard product that might be more cost effective for some applications. However, it would result in significantly higher chip and board counts.

9.8 FPGA Computing Machine

FPGA based computers allow their hardware resources to be reconfigured to fit the job at hand. They offer the possibility of combining the speed of custom hardware with the reprogrammability of software. For example, an FPGA based SAR processor could be configured as an FFT processor for the fast convolution operations, and then reconfigured as a resampling processor for the geometric correction operations. The circuit used in each case would be optimized for the required algorithm.

This section examines the feasibility of using FPGAs in SAR processing applications, and makes some predictions of the performance that can be expected from existing FPGA devices.

9.8.1 Standard Arithmetic

This section analyzes the performance achieved by FPGAs on DSP operations based on results reported in the literature.

In the analysis given in [9], the basic building block is a multiplier designed by Casselman. The 24-bit fixed point version of the multiplier requires 48 configurable logic blocks (CLB) in a Xilinx XC4000 series

device and produces a result in 16 clock cycles at 16 MHz. This gives 1 MOP/sec per multiplier. Given that the Xilinx XC4010 device has 400 CLBs, and assuming 25% overhead, results in about 6 MOP/sec per chip. The single precision floating point version of the multiplier requires 60 CLBs and produces a result in 16 cycles at 16 MHz. This gives about 5 MOP/sec per XC4010 chip. Somewhat higher performance could be obtained by using the largest available FPGA which is currently the XC4025 with 1024 CLBs.

In the results given in [6] for the SPLASH-2 hardware platform which uses 16 XC4010 FPGAs as the processing units, a 2D FFT implementation can process two 512×512 images per second. The wordlength is 8 bits. This gives an operation rate of 47 MOP/s. Only five FPGAs are used for the actual butterfly computations.

Both [9] and [6] give the idea that the order of magnitude of performance to be expected is about 5-10 MOP/s per 400-CLB chip for MAC-type operations with 8 to 24-bit wordlengths. This is a relatively low level of performance. A large number of chips is required to achieve performance levels higher than that available from general purpose DSPs. Current FPGAs do not have sufficient resources for the efficient implementation of standard arithmetic units like multipliers. It can be concluded that current FPGAs are not suitable for SAR processing if regular MAC structures are implemented.

9.8.2 Distributed Arithmetic

An interesting result with much higher performance is contained in a 16-tap 8-bit FIR filter macro for the XC4000 series devices produced by Xilinx [41]. The filter runs at 5.44 Msamples/sec and uses 67 CLBs. This gives an operation rate of 169 MOP/sec per filter. If multiple filters are programmed into an XC4010 with 400 CLBs, a single FPGA is able to realize about 800 MOP/s if 25% overhead is assumed. This impressive level of performance is much higher than a general purpose DSP, but is still lower than a special purpose DSP device like the ZR33288.

The Xilinx filter macro shows that it is possible to achieve very high performance with FPGAs for a specific application. The key to success in the case of the filter is the use of distributed arithmetic [92] [69] to avoid multiplications, which are expensive to implement in FPGAs.

The Xilinx macro has some serious limitations however:

- The word length is only eight bits.
- The coefficients are not variable. The FPGA must be reprogrammed to change them.

- The filter relies on the symmetry of linear phase FIRs: actually only eight unique coefficients are handled.

As a result, the design as it stands is not very practical. Table 9.5 shows how the filter scales as the design parameters are changed and how different approaches affect the scalability.

Table 9.5: Scaling the distributed arithmetic filter

Design Parameter Change	Approach	Result
increase filter length	in one filter section	area increases exponentially
	partition into multiple parallel filter sections	area increases linearly
	partition into multiple cascade filter sections	area increases linearly (but slower than for parallel sections)
		speed decreases linearly
increase wordlength	in one filter section	area increases linearly
		speed decreases linearly
	in duplicate sections	area increases linearly

The original Xilinx filter macro is able to achieve 800 MOP/s per chip with 8-bit real numbers. If the filter is adapted to handle 24-bit complex numbers, the performance per 400 CLB FPGA would be expected to decrease to a maximum of about 100 MOP/s. This is somewhat faster than most general purpose DSPs but is slower than the LH9124. When compared with commercial fixed-function filter chips like the ZR33288, the FPGA approach to time domain SAR processing exhibits

- lower performance,
- even more hardware design,
- a much more flexible architecture, and
- a range of wordlengths and number formats.

So far, the good performance obtained from distributed arithmetic has only been applied to filtering. While time domain processing can be used for SAR, it has a number of disadvantages as seen in Section 9.5. The key to making more general application of FPGAs to SAR processing is an efficient implementation of FFTs. FFTs can be implemented using distributed arithmetic using the same principles as are used in filters [59] [75] [90] [61] [91] [83]. The multipliers are again replaced with look-up tables and adders. However, considerably more resources are required in this case since

- support for complex arithmetic is needed,

- some kind of floating point support is probably also needed, and
- the number of twiddle factors required for longer FFT lengths make an external ROM necessary.

It is predicted that a 1k CFFT time of about 1 ms for 16-bit data is achievable in the largest FPGA currently available. This prediction is based on an analysis of reported FFT and filter implementations in FPGAs. This is equivalent to a processing rate of about 5 Mbutterflies/sec. This is in the same league as general purpose DSPs, but is slower than the fastest ones, and is only for a relatively short word length. Of course, higher performance can be achieved by using multiple FPGAs.

9.8.3 FPGA Conclusions

Some of the advantages of FPGA based SAR processing approaches are listed below.

- **Reconfigurability:** The hardware designs implemented in the FPGAs can be changed to adapt to different processing requirements.
- **Flexibility:** FPGA based designs can handle a wide range of algorithms, word lengths, number formats, etc.
- **Generality:** It is possible to build relatively general computing machines that are useful for other algorithms or applications.
- **Parallelism:** Much potential exists for fine grain parallelism.
- **Migration:** A clear migration path to ASICs exists.

Some disadvantages are:

- **Performance:** Performance will be mediocre.
- **Cost:** The development costs are likely to be high.
- **Risk:** FPGA based DSP is an unproven technology.

A possible solution to the problem of implementing arithmetic functions in FPGAs is to combine FPGAs with commercial arithmetic chips like multipliers and ALUs. In this case, the commercial chips would implement the arithmetic and the FPGAs the control logic. There are several potential problems with this approach:

- Commercial arithmetic chips are not much faster than general purpose DSPs.

- The greatest potential for speed-up in FPGAs is through the extensive use of parallelism. This leads to large chip counts if separate arithmetic chips are used.
- A primary advantage of FPGAs is their flexibility. This begins to be lost if commercial chips are used.

FPGA performance also depends on reprogramming time. Existing reconfigurable machines usually configure the FPGAs before processing starts. In DSP applications that make use of a number of different operations, it is desirable to reconfigure the FPGAs during processing. This requires the ability to reconfigure the FPGAs quickly. Until now FPGAs have had relatively long programming times, however, this is decreasing in more recent devices.

In order to improve the cost effectiveness of FPGA based computers, the development costs need to be reduced significantly. This will be facilitated by the development of

- standard FPGA based computing platforms,
- better design tools, and
- standard libraries.

FPGAs will undoubtedly play a large part in any custom hardware design due to their usefulness in implementing control logic. However, their role as computational devices is limited. FPGA based signal processing is not yet a mature enough technology to be cost effective. Higher density FPGAs, hardware platforms that support extensive fine grain parallelism, and powerful software tools are needed for FPGAs to be an attractive alternative to commercial DSP chips.

9.9 Architecture Examination Conclusions

This section summarizes the conclusions reached about each architecture and makes some comparisons between them. The degree to which each of the architectures meets the criteria set out in Section 3.5 is shown in Table 9.6. All of the architectures discussed in this section are listed in the table with the exception of the fine grain architecture. A design for a fine grain processor would have to be specified before comparisons can be made.

In the table, a scale of one to five is used, where a score of five is the most desirable and one the least. It should be emphasised that the scores are quite subjective and have no absolute meaning. The scores simply provide a means of comparing the architectures. The scores in Table 9.6 for each of the five criteria are discussed in the paragraphs that follow.

Table 9.6: Architecture trade-off

Architecture	Performance	Scalability	Flexibility	Development Cost	Repeat Cost
NOW	2	3	5	5	1
GP DSP	4	4	4	4	4
Vector DSP	5	4	3	3	4
Optimized DSP	5	4	2	1	4
Filter	4	4	2	2	4
VSA	5	4	4	3	4
FPGA	3	4	3	2	3

Performance

All of the architectures are capable of high performance. In the table, the scores for performance are based on the performance that can be expected per board. One of the fundamental trade-offs in these architectures is the one between the performance and the generality of the architecture. This is depicted in Figure 9.20.

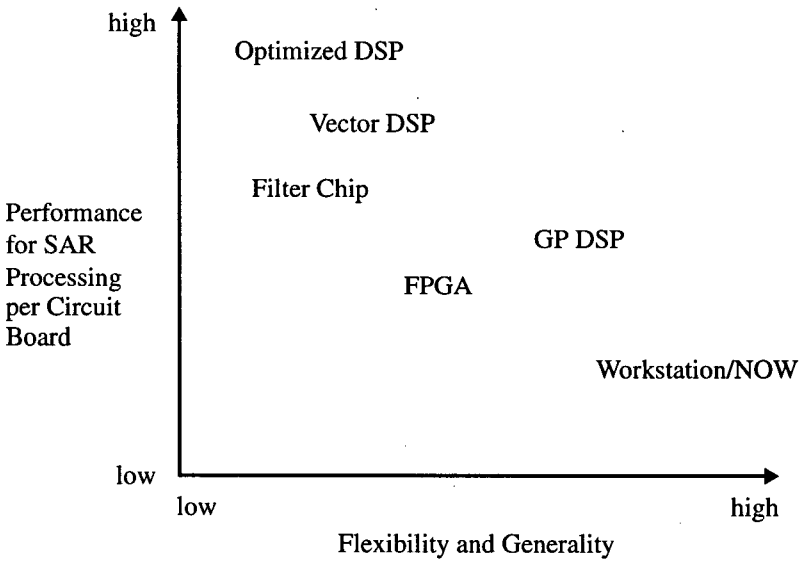


Figure 9.20: Performance/generality trade-off of processor architectures

Scalability

All architectures are easily scaled to higher performance levels. NOW is considered to be the least scalable because of the limited interconnect bandwidth and the high cost of additional PEs.

Flexibility

As can be seen from Figure 9.20, there is a large range of flexibility in the processor architectures, ranging from the extremely flexible NOW to much less flexible architectures like the filter chip approach.

Development cost and repeat cost

There is also a great deal of variation in the costs associated with each processor architecture. NOW has the lowest development cost but the highest repeat cost. The more specialized DSP based systems have the highest development cost and the lowest repeat cost.

The paragraphs that follow summarize the usefulness of each architecture for SAR processing focusing on their performance.

Network of workstations

NOW is a high performance solution best suited for situations where:

- Very complex processing algorithms are needed.
- A large amount of computation in addition to the basic SAR processing algorithm needs to be performed. This helps hide the problem of limited network bandwidth.
- A large amount of flexibility is needed. An example is if the same hardware is to be used for other purposes besides SAR.

General purpose DSP architecture

General purpose DSP architectures are a good compromise between flexibility and performance.

Vector DSP architecture

Vector DSP architectures offer the highest performance that is available with commercial ICs. The penalty in comparison to general purpose DSPs is less flexibility and more design work.

Optimized DSP

A custom optimized DSP approach must be taken if higher performance or integration than that offered by vector DSPs is required. An optimized DSP potentially offers the ultimate in performance for the highest development cost.

Digital filter chip architecture

Digital filter chip architectures can offer a significant performance advantage for a specific class of low to medium resolution processing scenarios. However, their development cost is higher and their flexibility is lower than for general purpose DSPs.

Fine grained architecture

Fine grained architectures were not found to be necessary for SAR processing except for certain situations where benefits can be realized by using existing coarse grain components in a fine grain manner. This will likely remain the case unless commercial products become available that give fine grain parallel architectures a cost advantage.

Sample heterogeneous architecture

A hybrid architecture that combines general purpose DSPs with a vector DSP was found to offer a performance advantage over general purpose DSPs at the expense of flexibility and development cost.

FPGA computing machine

FPGA based computers were found to offer no performance advantages for general SAR processing over other devices, and additionally carry the price of higher development cost. Currently, FPGAs are potentially useful for some specific portions of SAR processing where a commercial chip is not appropriate for some reason. Future advances in FPGA devices and systems will expand their usefulness.

10 Conclusions and Future Work

10.1 Conclusions

This thesis has examined processor architectures for SAR by looking at the three components of system design: algorithm, technology, and architecture.

Algorithm (Section 2)

An overview of SAR processing theory was given with an emphasis on deriving general performance and memory requirements for SAR processors. This overview attempted to fill a gap in the SAR processing literature by providing a concise summary of the equations that affect processor design. A selection of representative SAR algorithms was discussed and compared. The comparison is a more comprehensive extension of similar studies previously documented. A numerical analysis of their computational requirements supported the previously reported computational characterizations of these algorithms. FFTs were found to account for between 75% and 92% of the arithmetic operations required by the non-time domain algorithms. Since range-Doppler is representative of most common SAR algorithms, a more detailed model of this algorithm was developed for use in analyzing potential architectures.

The high computation requirements of most SAR processing situations necessitates the use of multiple processors. The options for partitioning SAR algorithms onto multiple processors were examined. The relative advantages of coarse and fine grain partitioning approaches for SAR processing were explored. A taxonomy of partitioning approaches was presented. The alternatives include horizontal and vertical partitioning and their combinations with various data partitioning schemes. The partitioning of the range-Doppler algorithm was modelled. Pure horizontal partitioning was found to introduce large inefficiencies due to the overlaps required between data blocks.

Technology (Section 5)

A survey was made of the current state of technology in the key areas that underlie SAR processor design.

The most important recent developments in processing devices include the increase in performance for DSP-type operations, the rise in on-chip integration, and the emergence of built-in support for multiprocessing. In order to provide a quantitative measure for comparison, FFT performance figures for various types of processing devices were collected and compared. Modern RISC processors were found to be equally as fast as DSPs at DSP operations like FFTs. However, DSP chips have other advantages that make them easier to use in high performance DSP applications. The highest performance available from commercial chips for general SAR operations is offered by vector DSPs. These devices have approximately an

order of magnitude faster FFT performance than general purpose DSPs. Despite the high performance figures, no existing single processing device is fast enough to support 1/10 real-time satellite SAR processing. Any high performance SAR processor must make use of some form of parallel processing.

Recent developments in RAM technology have much to offer to SAR processors. Increased memory capacity is an obvious advantage. The increase in DRAM speed for certain types of accesses brings the promise of higher processing speed, provided it is utilized properly. However, the lack of significant improvements in DRAM random access times leads to a need for careful memory systems design.

Many options exist for the interconnection networks of SAR processors. The most important development in interconnections has been the emergence of standard interconnections aimed specifically at multiprocessing DSP applications.

Architecture (Section 3,4,6,7,8,9)

As a starting point in the architecture design process, the importance of considering all aspects of the regular systems engineering process was highlighted. The requirements of high performance, scalability, flexibility, and cost effectiveness were chosen for the architecture candidates. They were selected to represent a balance of real world requirements. The remainder of the work was concerned with finding and understanding the architecture options for SAR processors that address these requirements.

The classification of SAR processor architectures developed in this report is depicted in Figure 10.1. The high level taxonomy of architectures is represented by the first level of decomposition in Figure 10.1. It differs from standard taxonomies in that it has been adapted to encompass existing and anticipated SAR processor architectures.

Possible architecture implementation alternatives were chosen. The relation of these alternatives to the high level taxonomy is shown in the second level of decomposition in Figure 10.1. These combine the best of existing technology with the most promising architectures. The suitability of each architecture implementation for SAR processing was analyzed. Three were identified for further study: networks of workstations, DSPs, and FPGAs. The other architectures were eliminated from further study. These architectures and the reasons they were ruled out are as follows: workstation (performance and scalability), accelerated general purpose computer (cost and scalability), supercomputer (cost), and custom algorithm specific processor (flexibility).

Before continuing with a more detailed look at SAR processor implementations, two essential ingredients of the architectures where examined by themselves: memory systems and interconnection networks.

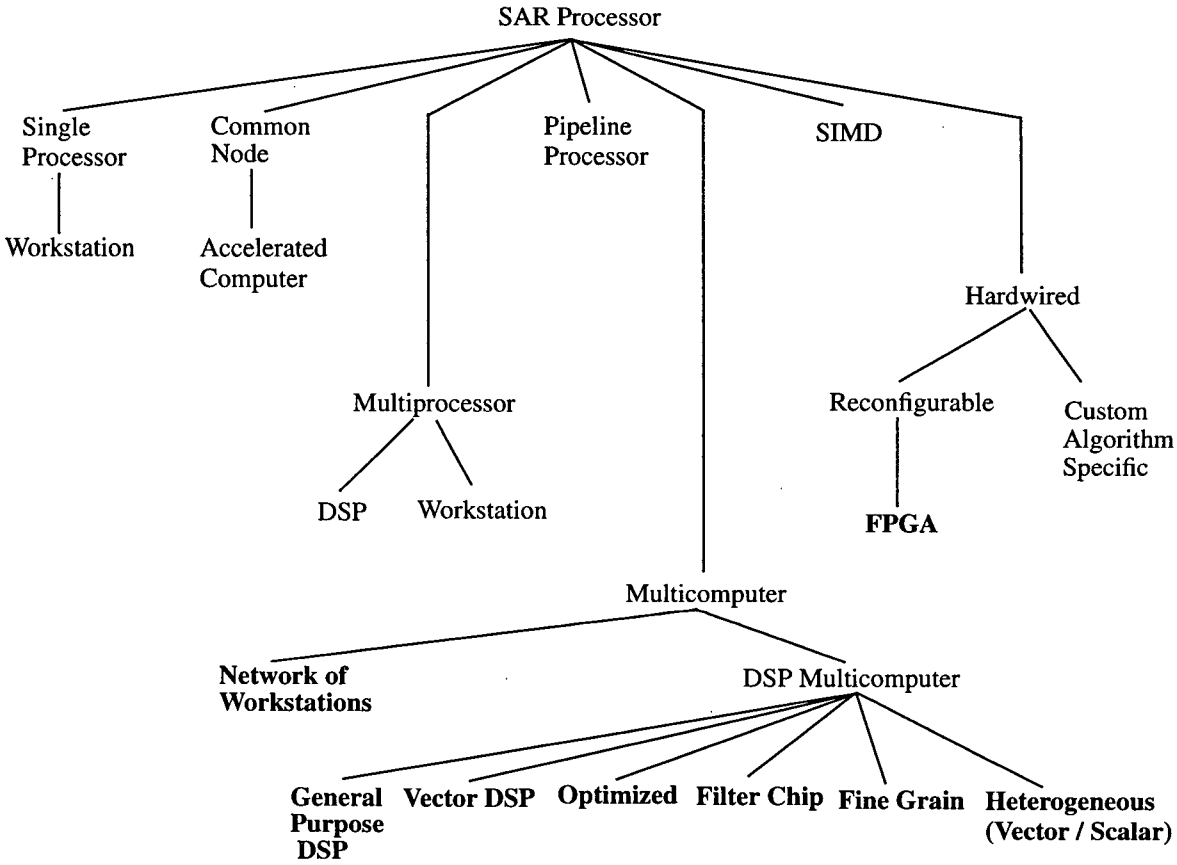


Figure 10.1: Classification of SAR processor architectures

Memory system design was identified as a major system design issue. The suitability of modern memory system design techniques to SAR was examined: large data path widths, caches, interleaving, and fast RAM devices. All were found to be useful if properly tuned to the memory access patterns of SAR. In particular, caches can potentially be detrimental if they are not flexible enough to handle the vector nature of SAR.

The most likely interconnection networks to be used in multicomputer SAR processors were identified and studied. The networks are the bus, mesh, and crossbar. Their corner turning performance was derived and compared. All the interconnection networks are suitable for SAR processors with relatively low numbers of PEs. For larger numbers of PEs, mesh and mesh-like interconnections offer the best performance.

A detailed analysis was performed of SAR processors based on networks of workstations, general purpose DSPs, vector DSPs, a proposed optimized DSP, digital filter chips, and FPGAs. Networks of workstations suffer from the problem of requiring much higher interconnection bandwidths than those offered by conventional LAN products. General purpose and vector DSPs provide good solutions for SAR processing. Approximately 16 Analog Devices SHARC general purpose DSPs would be necessary for 1/10 real time

processing. Approximately two LH9124 vector DSPs and a similar number of general purpose processors would be required. An optimized DSP based on the existing vector DSPs was proposed. The availability of such a device would significantly reduce the number of ICs required for a SAR processor and would eventually lead to the development of a single chip SAR processor. Digital filter chips and FPGAs where found to be less suitable for general purpose SAR processing.

The analysis demonstrated the strengths and weaknesses of commercial products. Commercial chips that address subsets of the needs of SAR processing are available. However, in general, it is difficult to find both good system level solution support and high performance on one chip. This observation led to the consideration of a heterogeneous approach that combines the advantages of various types of DSP devices. A design that combines a vector DSP with scalar general purpose DSPs was developed and its performance analyzed. It was found to offer a good balance between performance and flexibility. A design with one vector DSP (Sharp/Butterfly LH9124) and eight general purpose DSPs (Analog Devices SHARC) was found to be capable of handling 1/10 real-time processing. The provision of efficient memories and data transfer paths was of equal importance in the design as the selection of the processing units.

Each of the processor architectures considered meet the established requirements to varying degrees. All offer the potential for good performance and scalability. The trade-off between performance and flexibility was very evident in these architectures. In general, DSP based multicomputers were found to offer the best overall architecture for the SAR processing requirements considered. Future technological developments are expected to improve this further. Within a decade, single chip DSPs should be available that are able to handle all the computation required for the 1/10 real-time SAR processing scenario.

10.2 Future Work

This section identifies some possible areas for future work that extend the work presented in this thesis.

Network of workstations

Provided appropriate networking hardware and software is available, it would be interesting to develop an experimental NOW SAR processor system. Several powerful workstations with large memories and disks are required. These would probably already be on-hand. A high speed network with speeds at least as high as FDDI or ATM is necessary (i.e. with a bandwidth greater than 100 Mbps). The network hardware would most likely need to be procured, unless a fast network is already available due to some other related research. The SAR processing software could be an adaptation of existing SAR software that runs on a single workstation. The application should be made scalable so that it can use whatever number of workstations is available. An attempt should be made to bypass the overheads posed by the

networking protocols but still be as independent as possible of the underlying hardware. The software interface used for inter-processor communication needs to be chosen. Since the amount of coupling between the tasks running on different machines will need to be kept small, full message passing interfaces like PVM or MPI may not be necessary. However, if a portable interface like PVM or MPI is used, the disadvantage of higher overheads may be outweighed by the ability to more easily port the application to other platforms (e.g. to parallel machines like the SP-2).

General purpose DSP

The DSP multicomputer approach to building SAR processors can most easily be explored using a commercially available multi-DSP board that plugs into a standard low-cost computer platform like a PC or workstation. The board should contain multiple floating point DSPs, but can otherwise be a fairly low-end, inexpensive board. An example would be a PCI board that contains 2 to 4 SHARCs. Once again, the SAR processing software could be an adaptation of existing SAR software. The application would be split: the control and user interface would run on the host, while the signal processing portion would run on the DSPs. The challenge of the design would be to achieve high performance while leaving the software as portable as possible. A key decision that affects this is whether or not to use any high level programming toolkits offered by the DSP board vendor.

Optimized DSP

The design of an optimized DSP can be explored by modelling such a device using a behavioural HDL. Various design options and trade-offs can be explored. Existing models of standard building blocks or DSP cores can be used to accelerate this work. Synthesis tools can be used to verify the feasibility of the designs. It is quite possible that the designs produced will not be feasible for fabrication today. Various options for mapping the design to current technologies can be investigated (e.g. multiple chips, MCMs, etc.).

Digital filter chip processor

An experimental SAR processor based on digital filter chips can be designed and built based on the description given in Section 9.5.

Vector / scalar architecture

A version of VSA described in Section 9.7 can be designed and built. This project would have the allure of trying to put the maximum amount of performance for SAR on a single board. However, the development of such a board would be fairly expensive. A significant amount of support software would have to be written for both the VSA and the host computer. From the software side, this project is an extension of the general purpose DSP work described above.

FPGA computing machine

A large amount of research work remains to be done in the field of FPGA computing machines. The application of this field to SAR processing can be pursued at two levels. Designs for key portions of SAR processing can be described at a fairly high level and existing synthesis tools can be used to obtain implementations. Alternatively, new low-level microarchitectures that are suitable for FPGAs can be developed for the key operations in SAR. However, both of these approaches are really in a separate research area and will not offer competitive solutions for SAR processor designs for a number of years.

References

[1] Analog Devices, Inc. *1995 DSP/MSP Products Reference Manual*, 1995.

[2] A.H. Anderson, G.S. Downs, and G.A. Shaw. "RASSP benchmark-1 and -2: A preliminary assessment." Technical report, M.I.T. Lincoln Laboratory, 1995.

[3] Allan H. Anderson, Gary A. Shaw, and Chris T. Sung. "VHDL executable requirements." In *1st Annual RASSP Conference*, 1994.

[4] G. Leigh Anderson. "A stepwise approach to computing the multidimensional fast fourier transform of large arrays." *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-28(3):280, June 1980.

[5] Thomas E. Anderson, David E. Culler, David A. Patterson, and Others. "A case for NOW (networks of workstations)." Technical report, UC Berkeley, 1994.

[6] Peter M. Athanas and A. Lynn Abbott. "Real-time image processing on a custom computing platform." *IEEE Computer*, pages 16–24, February 1995.

[7] Stan Baker. "Reconfigurable computing." *Electronic Engineering Times*, Feb. 28 1994.

[8] J.R. Bennett, I.G. Cumming, and R.M. Wedding. "Algorithms for preprocessing of satellite SAR data." In *Proceedings ISPRS Commission II Symposium*, 1982.

[9] Neil W. Bergmann and J. Craig Mudge. "Comparing the performance of FPGA-based custom computers with general-purpose computers for DSP applications." In *1994 Workshop on FPGAs for Custom Computing*, 1994.

[10] Patrice Bertin, Didier Roncin, and Jean Vuillemin. "Introduction to programmable active memories." Technical Report 3, Digital Paris Research Laboratory, June 1989.

[11] P. Bjørstad, J. Cook, H. Munthe-Kaas, and T. Sørevik. "Implementation of a SAR processing algorithm on MasPar MP-1208." Technical Report 57, MasPar, 1991.

[12] Richard E. Blahut. *Fast Algorithms for Digital Signal Processing*. Addison-Wesley, Reading, Massachusetts, 1985.

[13] Benjamin S. Blanchard and Wolter J. Fabrycky. *Systems Engineering and Analysis*. Prentice Hall, 1990.

[14] Keith Boland and Apostolos Dollas. "Predicting and precluding problems with memory latency." *IEEE Micro*, pages 59–67, August 1994.

[15] B.A. Bowen and W.R. Brown. *Systems Design Volume II of VLSI Systems Design for Digital Signal Processing*. Prentice-Hall, 1985.

[16] Joe E. Brewer, L. Gray Miller, Ira H. Gilbert, Joseph F. Melia, Douglas Garde, and James E. DeMaris. "A monolithic processing subsystem." *IEEE Transactions on Components, Packaging, and Manufacturing Technology - Part B*, 17(3):310–317, Aug. 1994.

[17] Chappell Brown. "The programmable-logic assault." *Electronic Engineering Times*, pages 33,36, Feb. 8 1993.

[18] Paul Budnik and David J. Kuck. "The organization and use of parallel memories." *IEEE Transactions on Computers*, pages 1566–1569, December 1971.

[19] Dave Bursky. "Improved DSP ICs eye new horizons." *Electronic Design*, pages 69–82, Nov. 11 1993.

[20] Dave Bursky. "Advanced DRAMs deliver peak systems performance." *Electronic Design*, 43(16):42, Aug. 7 1995.

[21] C. Cafforio, C. Prati, and F. Rocca. "SAR data focussing using seismic migration techniques." *IEEE Transactions Aerospace Electronic Systems*, 27:199–207, 1991.

[22] Steven Casselman. "Virtual computing and the virtual computer." In *IEEE Workshop on FPGAs for Custom Computing Machines*. IEEE, 1993.

[23] Steven Casselman and Mike Thornburg. "Transformable computers." In *8th International Parallel Processing Symposium*. IEEE, 1994.

[24] CPU Info Center. http://infopad.eecs.berkely.edu/CIC/.

[25] Jinsong Chong and Hailiang Peng. "A distributed SAR processing system based on PVM framework." Technical report, Institute of Electronics, Chinese Academy of Sciences, 1995.

[26] Ian G. Cumming and John R. Bennett. "Digital processing of SEASAT SAR data." In *Record of the IEEE 1979 International Conference on Acoustics, Speech and Signal Processing*, Washington, D.C., April 2-4 1979.

[27] J. C. Curlander and R. N. McDonough. *Synthetic Aperture Radar – Systems and Signal Processing*. Wiley, 1991.

[28] Gordon Davidson. *Image Formation From Squint Mode Synthetic Aperture Radar Data*. PhD thesis, UBC, 1994.

[29] Angel L. DeCegama. *Parallel Processing Architectures and VLSI Hardware*. Prentice Hall, 1989.

[30] Marc A. D'Iorio, Joseph Lam, and John Whitman. "An overview of Canadian processing systems for Radarsat." In *IGARSS'93*, 1993.

[31] Jack J. Dongarra, Steve W. Otto, Marc Snir, and David Walker. "An introduction to the MPI standard." *to appear in Communications of the ACM*, 1995.

[32] J.O. Eklundh. "A fast computer method for matrix transposing." *IEEE Transactions on Computers*, pages 801–803, July 1972.

[33] J.O. Eklundh. "Efficient matrix transposition." In T.S. Huang, editor, *Two-Dimensional Digital Signal Processing II: Transforms and Median Filters*. Springer-Verlag, 1981.

[34] Charles Elachi, Tom Bicknell, Rolando L. Jordan, and Chialin Wu. "Spaceborne synthetic-aperture imaging radars: Applications, techniques, and technology." *Proceedings of the IEEE*, 70(10):1174–1209, Oct 1982.

[35] Tse-Yun Feng. "A survey of interconnection networks." *IEEE Computer*, pages 12–27, December 1981.

[36] J. Patrick Fitch. *Synthetic Aperture Radar.* Springer-Verlag, New York, 1988.

[37] M.J. Flynn. "Some computer organizations and their effectiveness." *IEEE Transactions on Computers*, C-21(9):948–960, Sept. 1972.

[38] John Gallant. "High-density PLDs." *EDN*, page 31, March 16 1995.

[39] A. Giri, V. Visvanathan, S.K. Nandy, and S.K. Ghoshal. "High speed digital filtering on SRAM-based FPGAs." In *7th International Conference on VLSI Design*, pages 229–232. IEEE, 1994.

[40] Maya Gokhale, William Holmes, Andrew Kopser, Sara Lucas, Ronald Minnich, Douglas Sweely, and Daniel Lopresti. "Building and using a highly parallel programmable logic array." *IEEE Computer*, pages 81–89, Jan 1991.

[41] G. Goslin. "Using Xilinx FPGAs to design custom digital signal processing devices." Technical report, Xilinx Publications, 1995.

[42] Brian K. Grant and Anthony Skjellum. "The PVM systems: An in-depth analysis and documenting study - concise edition." Technical report, Lawrence Livermore National Laboratory, 1992.

[43] Anshul Gupta and Vipin Kumar. "On the scalability of FFT on parallel computers." In *Third Symposium on the Frontiers of Massively Parallel Computation. Proceedings*, pages 69–74, New York, NY, 1990. IEEE.

[44] David B. Gustavson and Qiang Li. "Local-area multiprocessor: the scalable coherent interface." Technical report, SCIzzL, Santa Clara University, 1995.

[45] M.W. Haney. "Compact acousto-optic processor for synthetic aperture radar image formation." *Proceedings of the IEEE*, 82(11):1735–48, Nov. 1994.

[46] Johnathan C. Hardwick. "Porting a vector library: a comparison of MPI, Paris, CMMD and PVM." Technical report, School of Computer Science, Carnegie Mellon University, 1994.

[47] Haleh Hobooti. "Radiometric correction in range-SPECAN SAR processing." Master's thesis, UBC, 1995.

[48] Frederick R. Hood, III. "Reconfigurable hardware for digital signal processing algorithms." Master's thesis, University of Washington, 1993.

[49] Jeffrey I. Hutchings and Kent L. Gilson. "High performance digital signal processing using reconfigurable FPGA logic." In *The Proceedings of the 5th International Conference on Signal Processing Applications and Technology (ICSPAT'94)*, 1994.

[50] Barry Isenstein. "Scaling I/O bandwidth with multiprocessors." *Electronic Design*, pages 128–138, June 13 1994.

[51] Rich Jaenicke, Ken Linton, and Doug Williams. "VME, performance growing, gains support." *Electronic Engineering Times*, (847):54, May 8 1995.

[52] Raj Jain. *The Art of Computer Systems Performance Analysis*. John Wiley and Sons, 1991.

[53] Michael Y. Jin and Chialin Wu. "A SAR correlation algorithm which accomodates large-range migration." *IEEE Transactions on Geoscience and Remote Sensing*, GE-22(6):592–597, Nov 1984.

[54] Colin R. Johnson and Ron Wilson. "The 'virtual computer' gets boost from Altera." *Electronic Engineering Times*, page 1, Dec. 13 1993.

[55] R. Colin Johnson. "Computing 2000: The PC unbound." *OEM Magazine*, pages 88–9, March 1994.

[56] Asawaree Kalavade and Edward A. Lee. "A hardware-software codesign methodology for DSP applications." *IEEE Design & Test of Computers*, pages 16–28, September 1993.

[57] Duncan H. Lawrie. "Access and alignment of data in an array processor." *IEEE Transactions on Computers*, pages 1145–1155, December 1975.

[58] Markus Levy and James P. Leonard. "EDN's DSP-chip directory." *EDN*, pages 40–95, May 11 1995.

[59] Bede Liu and Abraham Peled. "A new hardware realization of high-speed fast fourier transformers." *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-23(6):543–547, Dec. 1975.

[60] Y. Luo, I.G. Cumming, and M.J. Yedlin. "Benchmarking a massively parallel computer by a synthetic aperture radar processing algorithm." In C.A. Brebbia and H. Power, editors, *Applications of Supercomputers in Engineering III*, pages 393–408. Computational Mechanics Publications and Elsevier Science Publishers, 1993.

[61] I.R. Mactaggart and M.A. Jack. "A single chip radix-2 FFT butterfly architecture using parallel data distributed arithmetic." *IEEE Journal of Solid-State Circuits*, SC-19(3):368–373, June 1984.

[62] Vijay K. Madisetti. *VLSI Digital Signal Processors - An Introduction to Rapid Prototyping and Design Synthesis*. IEEE Press, 1995.

[63] MDA. "Spectral analysis approach to the compression of linear FM signals." Technical Report ESTEC Contract 3998/79/NL/HP(SC), MDA, 1979.

[64] MDA. "Design concepts for the RadarSat SAR processor architecture." Technical Report RT-TN-50-3295, MDA, 1991.

[65] MDA. "Survey of SAR processing algorithms for the Radarsat SAR processor." Technical Report RT-RP-50-3264, MDA, 1991.

[66] MDA. "RSARP computing platform evaluation." Technical Report RZ-TN-50-4403, MDA, 1992.

[67] MDA. "IRIS X2C system segment design manual." Technical Report IR-MA-50-5064, MDA, 1993.

[68] MDA. "Proposal for the development of the Radarsat Fastscan processor." Technical Report 01-1453, MDA, 1993.

[69] Les Mintzer. "Mechanization of digital signal processors." In Douglas F. Elliott, editor, *Handbook of Digital Signal Processing Engineering Applications*. Academic Press, Inc., 1987.

[70] Richard Nass. "A designer's guide to computer buses." *Electronic Design*, 41(24):129, November 22 1993.

[71] National Semiconductor. *QuickRing Competitive Comparison*, 1995.

[72] Yoshiaki Nemoto, Hideo Nishino, Makoto Ono, Hitoshi Mizutamari, Katsuhiko Nishikawa, and Kaoru Tanaka. "Japanese Earth Resources Satellite-1 synthetic aperture radar." *Proceedings of the IEEE*, 79(6):800–809, June 1991.

[73] Natawut Nupairoj and Lionel M. Ni. "Performance evaluation of some MPI implementations on workstation clusters." Technical report, Dept. of Computer Science, Michigan State University, 1994.

[74] List of FPGA-Based Computing Devices. http://www.io.com/ guccione/HW_list.html.

[75] Abraham Peled and Bede Liu. *Digital Signal Processing: Theory, Design, and Implementation*. John Wiley and Sons, 1976.

[76] Betty Prince. *Semiconductor Memories*. Wiley, 1991.

[77] R. Keith Raney, Anthony P. Luscombe, E. J. Langham, and Shabeer Ahmed. "RADARSAT." *Proceedings of the IEEE*, 79(6):839–849, June 1991.

[78] R.K. Raney, H. Runge, R. Bamler, I. Cumming, and F. Wong. "Precision SAR processing using chirp scaling." *IEEE Transactions on Geoscience and Remote Sensing*, 32(4):786–799, July 1994.

[79] M. Sack, M.R. Ito, and I.G. Cumming. "Application of efficient linear FM matched filtering algorithms to synthetic aperture radar processing." *IEE Proceedings*, 132 Pt. F(1):45, February 1985.

[80] Sharp Corp. *LH9124 Digital Signal Processor Data Sheet*, 1993.

[81] G.A. Shaw, J.C. Anderson, and V.K. Madisetti. "Assessing and improving current practice in the design of application-specific signal processors." Technical report, RASSP, 1994.

[82] Michael R. Smith. "How RISCy is DSP?" *IEEE Micro*, pages 10–23, December 1992.

[83] S.G. Smith and P.B. Denyer. "Efficient bit-serial complex multiplication and sum-of-products computation using distributed arithmetic." In *Proc. 1986 International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2203–2206, 1986.

[84] Texas Instruments, Inc. *TMS320C4x User's Guide*, 1991.

[85] Texas Instruments, Inc. *Parallel Processing with the TMS32C4x*, 1994.

[86] Texas Instruments, Inc. *TMS320C8x (MVP) Online Reference CD-ROM*, 1995.

[87] Kiyo Tomiyasu. "Tutorial review of synthetic-aperture radar (SAR) with applications to imaging the ocean surface." *Proceedings of the IEEE*, 66(5):563–583, May 1978.

[88] David E. Van Den Bout, Joseph N. Morris, Douglas Thomae, Scott Labrozzi, Scot Wingo, and Dean Hallman. "AnyBoard: An FPGA-based, reconfigurable system." *IEEE Design and Test of Computers*, pages 21–30, Sept. 1992.

[89] Thomas C. Waugh. "Field programmable gate array key to reconfigurable array outperforming supercomputers." In *IEEE 1991 Custom Integrated Circuits Conference*. IEEE, 1991.

[90] S.A. White. "A simple FFT butterfly arithmetic unit." *IEEE Transactions on Circuits and Systems*, CAS-28(4):352–355, April 1981.

[91] S.A. White. "Results of a preliminary study of a novel IC arithmetic unit ofr an FFT processor." In *Proc. 18th Asilomar Conference on Circuits, Systems, and Computers*, pages 67–71, 1984.

[92] Stanley A. White. "Applications of distributed arithmetic to digital signal processing: A tutorial review." *IEEE ASSP Magazine*, pages 4–19, July 1989.

[93] C. Wu, K.Y. Liu, and M. Jin. "Modeling and a correlation algorithm for spaceborne SAR signals." *IEEE Transactions on Aerospace and Electronic Systems*, AES-18:563–575, 1982.

[94] Vojin Zivojnovic. "Benchmarks benediction." *Electronic Engineering Times*, (819):39, October 17 1994.

[95] B. Zuerndorfer and G.A. Shaw. "SAR processing for RASSP application." Technical report, MIT Lincoln Laboratory, 1994.

Publications Derived from this Thesis

Peter G. Meisl, Mabo R. Ito, and Ian G. Cumming. "Parallel synthetic aperture radar processing on workstation networks." *Proceedings of 10th International Parallel Processing Symposium*, pp. 716-723, IEEE, Honolulu, Hawaii, April 15-19, 1996.

Peter G. Meisl, Mabo R. Ito, and Ian G. Cumming. "Parallel Processors for Synthetic Aperture Radar Imaging." To appear in *Proceedings of 1996 International Conference on Parallel Processing*, Bloomingdale, IL, Aug. 12-16, 1996.

A Model of Range-Doppler Algorithm

The range-Doppler algorithm described in Section 2.5.4 was modelled using a Mathcad document. The key portions are shown in Figures A.1, A.2, and A.3. The amount of computation required by each bubble in the flow diagram of Figure 2.17 is shown and then summed to find the total amount of computation. Before and after each step (or implicit step) that changes the size of the data array, the total amount of memory required is shown.

Derived Parameters

Frame time: $Ftime := \frac{N_{az} - L_{az}}{PRF}$ $Ftime = 2.599$ sec

Output azimuth dimension $FinalNAZ := floor\left[OVSM \cdot \left(N_{az} - L_{az} - N_{res}\right)\right]$ $FinalNAZ = 4883$ samples

Output range dimension $FinalNR := floor\left[OVSM \cdot \left(N_r - L_r - N_{rcmc} - N_{res}\right)\right]$ $FinalNR = 7537$ samples

---

Data Volume on Input

$$Data_1 := \left(N_{az} - L_{az}\right) \cdot N_r \cdot 2 \cdot \frac{N_{bits}}{2^{20} \cdot 8} \qquad Data_1 = 160.034 \text{ MB}$$

1.0 I/Q Balance

$$RD_{10} := 3 \cdot \left(N_{az} - L_{az}\right) \cdot RVS\left(N_r\right) \qquad RD_{10} = 6.293 \cdot 10^7 \text{ OPs}$$

2.0 Range FFT

$$RD_{20} := \left(N_{az} - L_{az}\right) \cdot FFT1d\left(N_{rfft}\right) \qquad RD_{20} = 1.862 \cdot 10^9 \text{ OPs}$$

3.0 Range Multiply

$$RD_{30} := \left(N_{az} - L_{az}\right) \cdot CVM\left(N_{rfft}\right) \qquad RD_{30} = 1.718 \cdot 10^8 \text{ OPs}$$

4.0 Range IFFT

$$RD_{40} := \left(N_{az} - L_{az}\right) \cdot FFT1d\left(N_{rfft}\right) \qquad RD_{40} = 1.862 \cdot 10^9 \text{ OPs}$$

Data Volume after Range Compression

$$Data_2 := \left(N_{az} - L_{az}\right) \cdot \left(N_r - L_r\right) \cdot 2 \cdot \frac{N_{bits}}{2^{20} \cdot 8} \qquad Data_2 = 144.031 \text{ MB}$$

Data Volume before Azimuth Compression

$$Data_3 := \left(N_r - L_r\right) \cdot N_{az} \cdot 2 \cdot \frac{N_{bits}}{2^{20} \cdot 8} \qquad Data_3 = 168.75 \text{ MB}$$

5.0 Azimuth FFT

$$RD_{50} := \left(N_r - L_r\right) \cdot FFT1d\left(N_{az}\right) \qquad RD_{50} = 1.327 \cdot 10^9 \text{ OPs}$$

6.0 RCMC Shift and Interpolation

$$RD_{60} := N_{az} \cdot \left(N_r - L_r - N_{rcmc}\right) \cdot Inter\left(N_{rcmc}\right) \qquad RD_{60} = 6.626 \cdot 10^8 \text{ OPs}$$

7.0 Azimuth reference function multiply

$$RD_{70} := \left(N_r - L_r - N_{rcmc}\right) \cdot CVM\left(N_{az}\right) \qquad RD_{70} = 1.325 \cdot 10^8 \text{ OPs}$$

8.0 Azimuth IFFT

$$RD_{80} := \left(N_r - L_r - N_{rcmc}\right) \cdot FFT1d\left(N_{az}\right) \qquad RD_{80} = 1.325 \cdot 10^9 \text{ OPs}$$

Data Volume after Azimuth Compression

$$Data_4 := \left(N_r - L_r - N_{rcmc}\right) \cdot \left(N_{az} - L_{az}\right) \cdot 2 \cdot \frac{N_{bits}}{2^{20} \cdot 8} \qquad Data_4 = 143.817 \text{ MB}$$

Figure A.1: Computational analysis of range-Doppler algorithm (Part 1 of 3)

9.0 Azimuth Resampling

9.1 Apply Interpolators

$RD91 := (N_r - L_r - N_{rcmc}) \cdot FinalNAZ \cdot Inter(N_{res})$ $RD91 = 7.899 \cdot 10^8$ OPs

9.2 Generate Indices

$RD921 := FinalNAZ \cdot 1$ $RD921 = 4883$ OPs

$RD922 := FinalNAZ \cdot Round$ $RD922 = 2.442 \cdot 10^4$ OPs

$RD92 := RD921 + RD922$ $RD92 = 2.93 \cdot 10^4$ OPs

Total: $RD_{90} := RD91 + RD92$ $RD_{90} = 7.899 \cdot 10^8$ OPs

---

Data Volume after Azimuth Resampling

$Data_5 := (N_r - L_r - N_{rcmc}) \cdot FinalNAZ \cdot 2 \cdot \frac{N_{bits}}{2^{20} \cdot 8}$ $Data_5 = 200.875$ MB

10.0 Range Resampling

10.1 Apply Interpolators

$RD101 := FinalNR \cdot FinalNAZ \cdot Inter(N_{res})$ $RD101 = 1.104 \cdot 10^9$ OPs

10.2 Generate Indices

$RD1021 := 2 \cdot FinalNR \cdot 1$ $RD1021 = 1.507 \cdot 10^4$ OPs

$RD1022 := 2 \cdot FinalNR \cdot Round$ $RD1022 = 7.537 \cdot 10^4$ OPs

$RD102 := RD1021 + RD1022$ $RD102 = 9.044 \cdot 10^4$ OPs

Total: $RD_{100} := RD101 + RD102$ $RD_{100} = 1.104 \cdot 10^9$ OPs

---

Data Volume after Range Resampling

$Data_6 := FinalNR \cdot FinalNAZ \cdot 2 \cdot \frac{N_{bits}}{2^{20} \cdot 8}$ $Data_6 = 280.786$ MB

11.0 Detection

$RD_{110} := 3 \cdot FinalNR \cdot FinalNAZ$ $RD_{110} = 1.104 \cdot 10^8$ OPs

---

Data Volume

$Data_7 := FinalNR \cdot FinalNAZ \cdot \frac{N_{bits}}{2^{20} \cdot 8}$ $Data_7 = 140.393$ MB

12.0 Calculate Range Filter and SRC

$RD_{120} := 0$ $RD_{120} = 0$ OPs

13.0 Calculate Doppler Centroid

$RD_{130} := 0$ $RD_{130} = 0$ OPs

Figure A.2: Computational analysis of range-Doppler algorithm (Part 2 of 3)

14.0 Calculate RCM Shifts and Indices

14.1 Calculate RCM

$RD141 := N_{az} \cdot (N_r - L_r - N_{rcmc}) \cdot 6$ $\quad RD141 = 1.325 \cdot 10^8$ OPs

14.2 Round for Integer Shift

$RD142 := (N_r - L_r - N_{rcmc}) \cdot N_{az} \cdot Round$ $\quad RD142 = 1.104 \cdot 10^8$ OPs

14.3 Calculate Interpolator indices

$RD1431 := (N_r - L_r - N_{rcmc}) \cdot N_{az} \cdot Round$ $\quad RD1431 = 1.104 \cdot 10^8$ OPs

$RD1432 := (N_r - L_r - N_{rcmc}) \cdot RVS(N_{az})$ $\quad RD1432 = 2.209 \cdot 10^7$ OPs

$RD143 := RD1431 + RD1432$ $\quad RD143 = 1.325 \cdot 10^8$ OPs

Total $RD_{140} := RD141 + RD142 + RD143$ $\quad RD_{140} = 3.755 \cdot 10^8$ OPs

15.0 Calculate Ambiguity

$RD_{150} := 0$ $\quad RD_{150} = 0$ OPs

16.0 Calculate Azimuth Filter

16.1 Calculate Azimuth Filter in Time

$RD161 := \frac{1}{n_u} \cdot (N_r - L_r - N_{rcmc}) \cdot L_{az} \cdot 20$ $\quad RD161 = 6.47 \cdot 10^7$ OPs

16.2 FFT

$RD162 := \frac{1}{n_u} \cdot (N_r - L_r - N_{rcmc}) \cdot FFT1d(N_{az})$ $\quad RD162 = 1.325 \cdot 10^9$ OPs

16.3 Frequency Weight Filter

$RD163 := \frac{1}{n_u} \cdot [(N_r - L_r - N_{rcmc}) \cdot CVRVM(N_{az})]$ $\quad RD163 = 4.417 \cdot 10^7$ OPs

16.4 Phase compensation

$RD1641 := \frac{1}{n_u} \cdot (N_r - L_r - N_{rcmc}) \cdot N_{az} \cdot 10$ $\quad RD1641 = 2.209 \cdot 10^8$ OPs

$RD1642 := \frac{1}{n_u} \cdot (N_r - L_r - N_{rcmc}) \cdot CVM(N_{az})$ $\quad RD1642 = 1.325 \cdot 10^8$ OPs

$RD164 := RD1641 + RD1642$ $\quad RD164 = 3.534 \cdot 10^8$ OPs

Total $RD_{160} := RD161 + RD162 + RD163 + RD164$ $\quad RD_{160} = 1.787 \cdot 10^9$ OPs

Statistics:

$i := 10, 20 .. 160$

Grand Total $RD_0 := \sum_i RD_i$ $\quad RD_0 = 1.157 \cdot 10^{10}$ OPs

Operation Rate: $MOps := \frac{RD_0}{Ftime \cdot 10^6}$ $\quad MOps = 4452.245$ MOP/s

Figure A.3: Computational analysis of range-Doppler algorithm (Part 3 of 3)