Realtime Synthetic Aperture Radar Processing on the RACE® Multicomputer

Application Note 203.0

TARGET AUDIENCE

System design engineers who are tasked with designing realtime SAR processing systems using the RACE Multicomputer.

Software engineers who desire to understand SAR signal processing algorithms and their implementation on the RACE Multicomputer.

ABSTRACT

This note develops relationships for predicting the memory and computational requirements of realtime SAR processing as a function of radar application parameters such as resolution, swath width, platform velocity, radar wavelength and range, for the simplest case of the range/Doppler stripmap algorithm. No prior knowledge of SAR processing is assumed. Using these general results to determine the processing requirements of a given application, the process of mapping the range/Doppler stripmap SAR processing algorithm onto a RACE Multicomputer is described. This mapping process includes distributing the total processing load among the individual CEs in the multicomputer so as to balance the computational load over all CEs as equally as possible and determining the memory required by each CE. An example of this mapping process is then illustrated for the case of a typical SAR scenario.

RACE hardware features that are especially relevant to SAR processing include efficient interprocessor communications and a chained-DMA capability on each CE to facilitate "corner-turning" operations. Corresponding software features include Mercury's Parallel Applications System (PAS) to facilitate the distribution of the SAR processing algorithm over multiple processors.

The application note is presented in overhead format with speaker notes.

Authored by:
Thomas Einstein, Systems Engineering

REVISION HISTORY	
Revision #	**Description**
203.0	Creation

Synthetic Aperture Radar Processing on the *RACE* Multi-computer

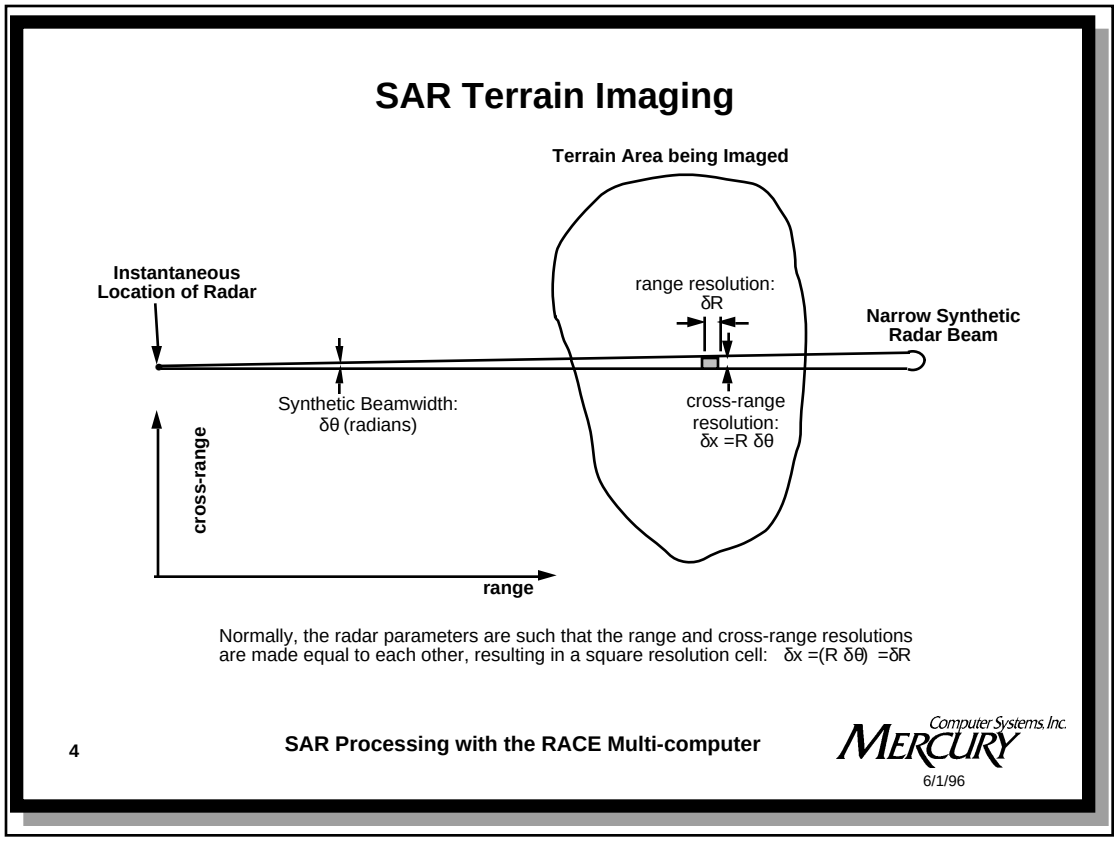
Outline of Presentation

» **Application: Synthetic Aperture Radar (SAR)**

- **Static Description: SAR Signal Processing**
- **Dynamic Description: Real-Time SAR Signal Processing**
- **Resource Requirements: Processing and Memory**
- **Mapping SAR Computations onto *RACE***

Introduction to SAR (Synthetic Aperture Radar)

- **Generates High-resolution aerial photograph-like images**
  - Image Resolution ranges from (50 x 50) m down to (0.5 x 0.5) m
- **Applications (both Military and Commercial)**
  - Ground Surveillance (all weather; day or night)
  - Terrain Mapping " " " "
  - Object Imaging (Inverse SAR / ISAR)
- **Requires relative motion between radar platform and area being imaged.**
  - Airborne SAR
  - Satellite SAR (imaging of earth and planetary terrain)
  - ISAR (imaging of satellites by ground radars)



This slide illustrates the use of Radar to do terrain imaging.

The key is to use a radar that has fine resolution in both the range and cross-range dimensions.

Fine range resolution is relatively straight-forward to achive.

However the required cross-range resolution usually requires a beam-width narrower than what can be achieved with a practically-sized physical antenna. The required antenna aperture is therefore generated synthetically by moving the physical antenna -thus the name Synthetic Aperture.

The Synthetic Aperture Image Resolution

- **Coordinates of SAR Image are Range and Cross-Range**
- **Coordinate Pixel Resolution**
  - **Range:** $\delta R = \frac{c}{2}\tau = \frac{c}{2B}$, m τ = compressed pulsewidth, secs; B = pulse bandwidth, Hz
  - **Cross-Range / Azimuth**

    λ = radar wavelength, m; L_e = effective antenna aperture, m

    Azimuth: $\delta\theta \approx \frac{\lambda}{2L_e}$, radians

    Cross-Range: $\delta x = R\delta\theta \approx \frac{R\lambda}{2L_e}$, m
- **Fine Cross-Range Resolution requires a large aperture, L_e**

This slide illustrates the relationship between the radar parameters and the critical SAR parameters: range resolution, δR and cross-range resolution, δx.

δR is inversely proportional to the radar pulse bandwidth, B.

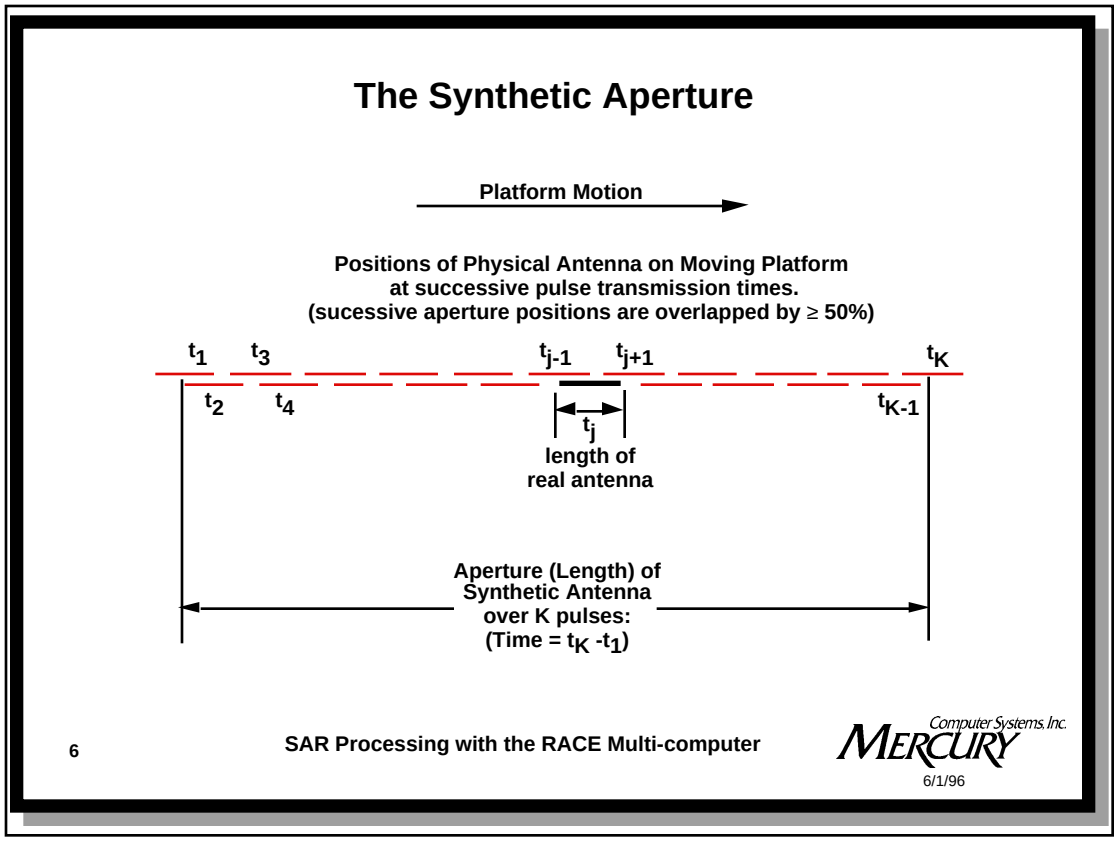
δx is inversely proportional to the radar's effective antenna aperture, L_e.

Usually, for SAR, the radar parameters B and L_e are chosen such that δR =δx.

For fine resolution it is therefore desirable that both B and L_e be large.

Large B can easily be achieved by transmission of "chirp" pulses.

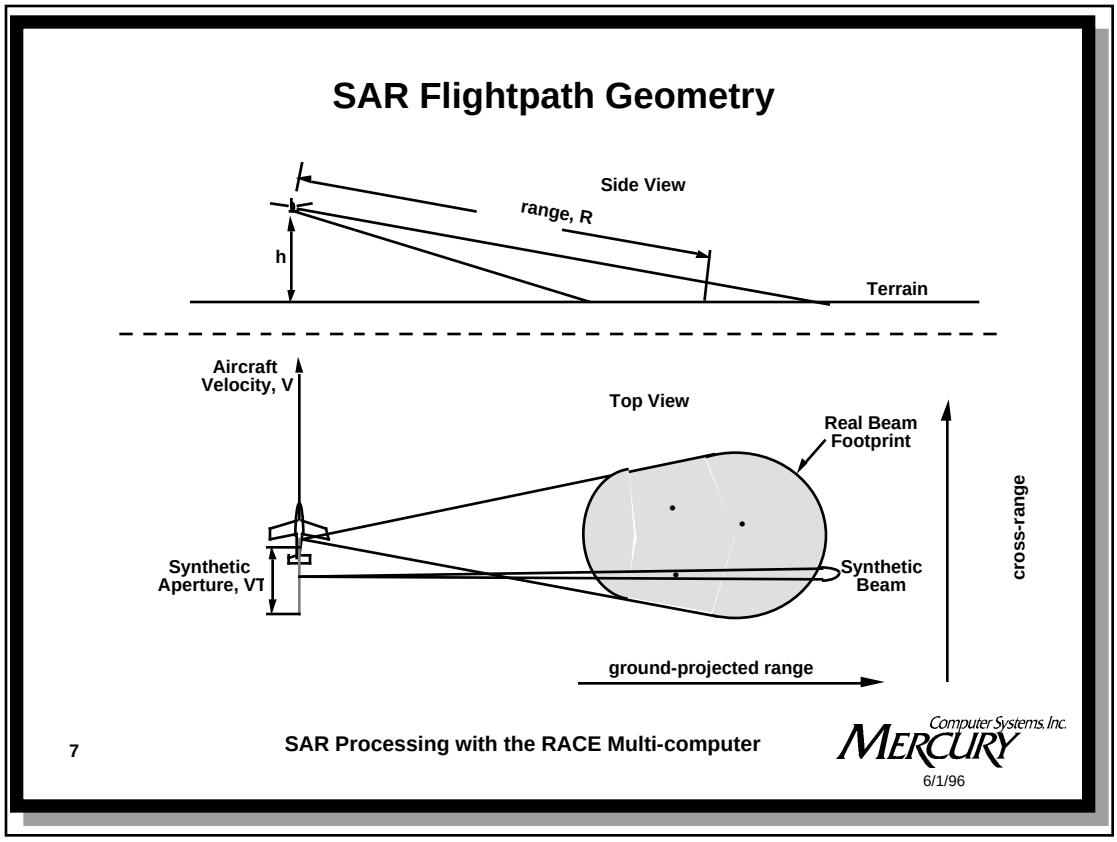
A sufficiently large value of effective antenna aperture, L_e, will generally exceed the physical limits imposed by the air- or spaceborne radar platform. Hence, this requisite value of L_e must be realized synthetically.



This slide illustrates the long synthetic antenna aperture that can be generated by the motion of the radar platform.

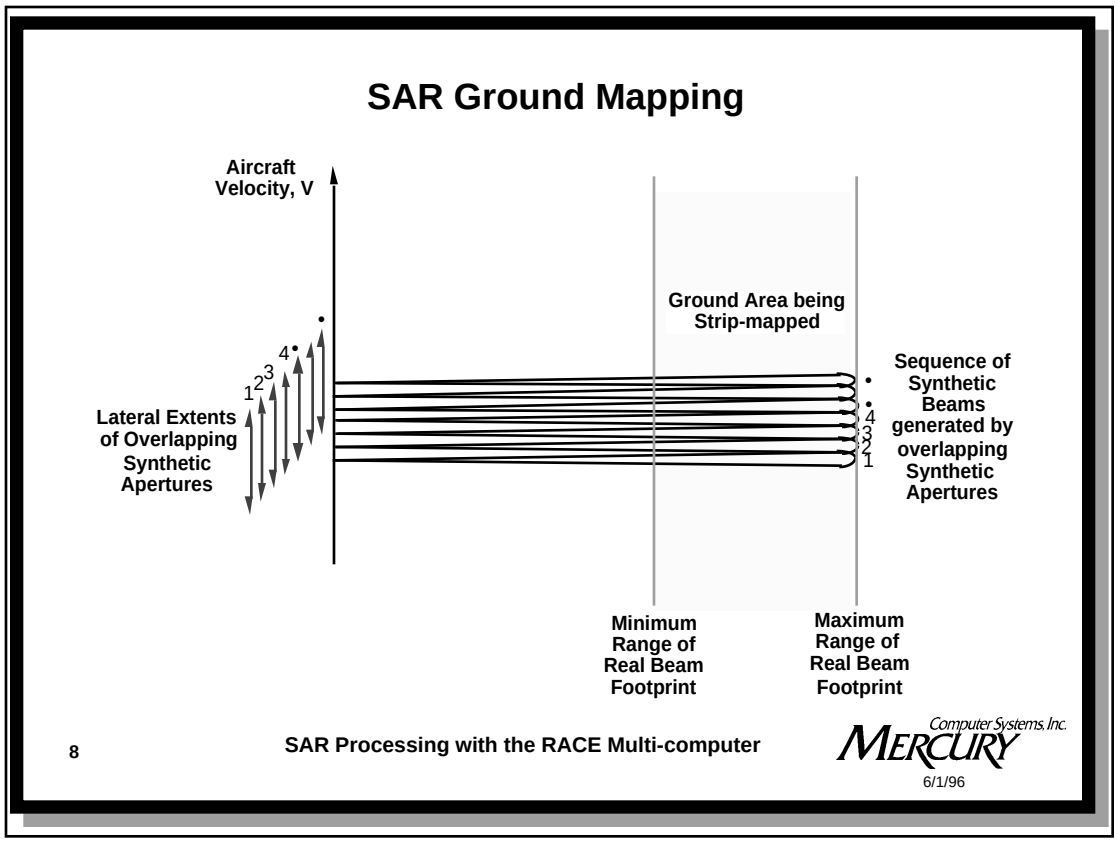
The length of the resultant synthetic antenna is generally hundreds to thousands of times longer than that of the physical antenna, resulting in synthetic beams that are hundreds to thousands times narrower than the radiated beamwidth of the physical antenna.

Each radar pulse return may be considered a sample point along the synthetic aperture. Nyquist sampling considerations require that the distance traversed by the radar platform during one pulse repitation interval not exceed one-half the length of the physical antenna, in order to avoid aliasing.



This slide illustrates the side and top views of the flight path geometry, illustrating the area of terrain illuminated by the radar's radiated beamwidth.

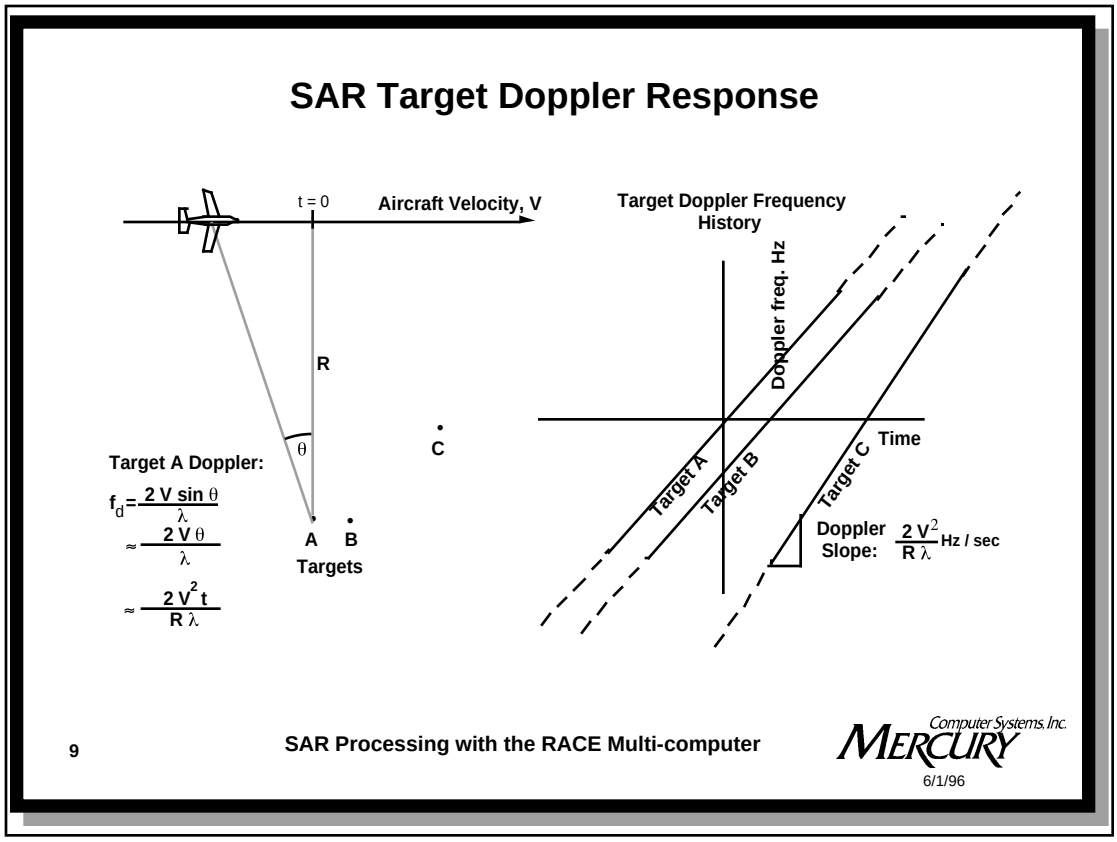
Both the physical and synthetic beam patterns are illustrated. Note the narrowness of the synthetic beam pattern compared to that of the radiated physical beam pattern



This slide illustrates how successive synthetic beams are generated every radar pulse return, during the motion of the radar platform, to produce a high resolution ground map of the illuminated range swath of terrain that is being imaged.

The region of terrain being imaged is a strip of width ΔR, parallel to the radar's flight path. Hence this type of SAR is called "strip-mapping".

Note that successive synthetic beams are formed from overlapping synthetic apertures



This slide illustrates the operation of SAR from a Doppler frequency point of view, rather than from an antenna-theoretic point of view.

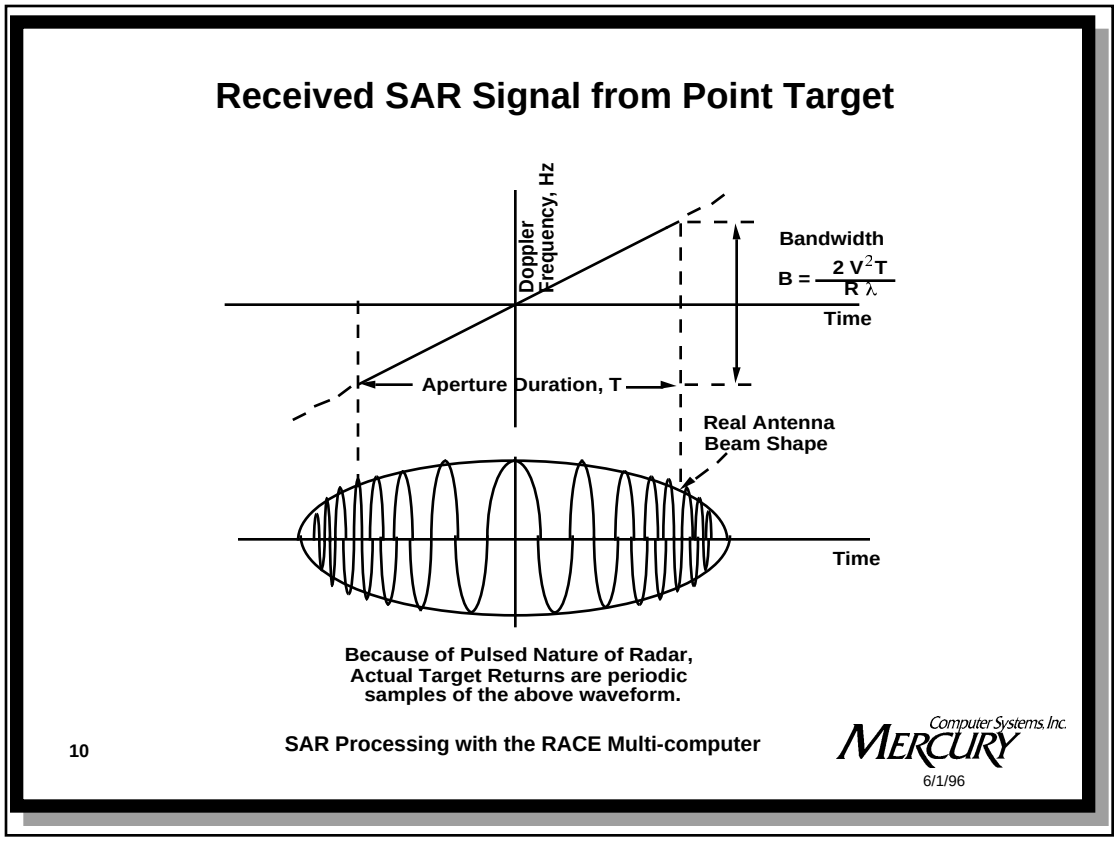
The two points of view can be shown to be equivalent. However the Doppler point of view facilitates a discussion of SAR signal processing.

As the aircraft flies by a point target (i.e. a single resolution cell) on the ground, the return from that cell has a Doppler frequency (due to aircraft motion) that varies approximately linearly with time over the duration of the synthetic aperture flight time.

The frequency slope of this return signal varies as the square of the platform velocity and inversely as the range to the target.

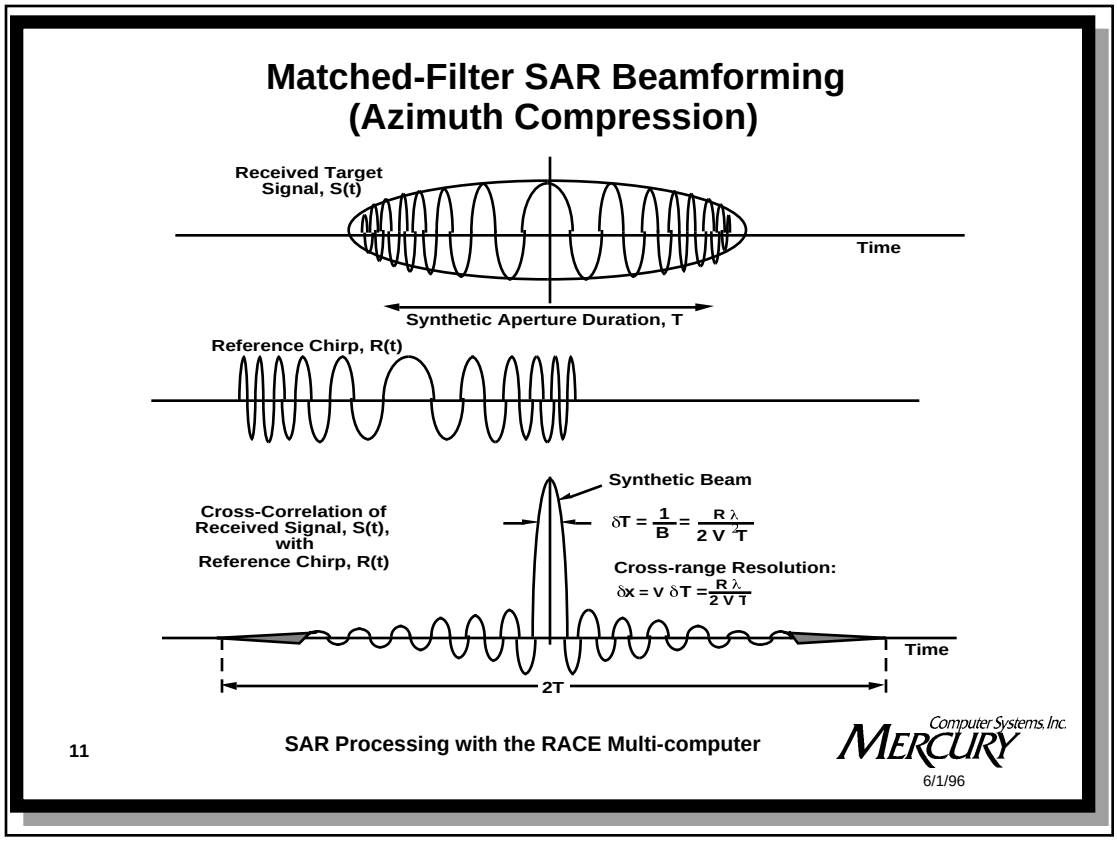
It is assumed that the variation in range from the radar to the given target, over the duration of the synthetic aperture, is less than the radar's range resolution so that the target will remain within the confines of a give range resolution cell for the duration of the synthetic aperture flight time.

If this assumption is violated, a more complicated processing algorithm (e.g."range migration" processing) than that to be presented must be used.



This slide shows the swept frequency waveform of the signal received from a point target (i.e. a single resolution cell).

The amplitude of this waveform tapers off at the edges of the synthetic aperture in accordance with the attenuation pattern (beamshape) of the real antenna.

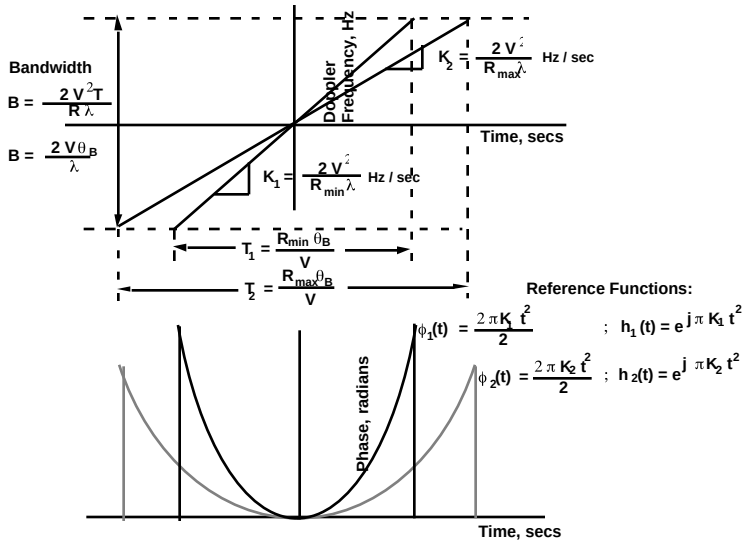


This slide illustrates how the synthetic beam pattern is generated by correlating the received signal (last slide) with a pre-stored reference waveform.

Note that the cross-range resolution, δx, of the resulting synthetic beam pattern is exactly equal to that that would have been obtained using the antenna theoretic approach.

This prestored reference waveform is a function of both target range and platform velocity. In practice, the image range swath is subdivided into segments and a different reference function is used for each range segment.

SAR Reference Functions vs Range



SAR Processing with the RACE Multi-computer

Computer Systems Inc. MERCURY

6/1/96

This slide reviews how SAR reference functions vary as a function of range.

The top part of the slide shows a representation of the refernce function in terms of instantaneous frequency vs time.

However, mathematically (i.e. in software) SAR reference functions are represented in terms of phase angle (or equivalently, a complex unit vector) vs time. Since phase angle is the time-integral of frequency, the phase functions corresponding to a linear variation in frequency will be quadratic functions of time, as illustrated.

SAR Time-Bandwidth Product

Bandwidth: $B = \frac{2 V^2 T}{R \lambda}$

Max. Synthetic Aperture Time: $T = \frac{R \theta_B}{V} \rightarrow B = \frac{2 V \theta_B}{\lambda}$

Time-Bandwidth Product: $BT = \frac{2 V T \theta_B}{\lambda} = \frac{2 R \theta_B^2}{\lambda}$

Synthetic Beamwidth: $\delta\theta = \frac{\lambda}{2 V T} = \frac{\lambda}{2 R \theta_B} = \frac{\delta_x}{R}$

Time-Bandwidth Product: $BT = \frac{\theta_B}{\delta\theta} = \frac{R \lambda}{2 \delta_x^2}$; **Azimuth Compression Ratio**

But, to prevent aliasing the minimum prf must equal the maximum Doppler spread across the real beamwidth θ_B

$prf_{min} \geq B = \frac{2 V \theta_B}{\lambda}$

Hence: $BT = T\, prf_{min} = N_{min}$ **Minimum Number of pulses in the Aperture and corresponding reference function samples.**

Reference Function: $\phi(t) = -\pi K t^2$; $K = \frac{2 V^2}{R \lambda}$

Reference Function Samples: $\phi_i = \phi(i \Delta t) = -\frac{2\pi}{R \lambda} \left(\frac{V}{prf}\right)^2 i^2$; $i = 1,2,\ldots, N \geq N_{min}$

13

SAR Processing with the RACE Multi-computer

Computer Systems Inc. MERCURY

6/1/96

This slide introduces the concept of the dimensionless parameter: the Time-Bandwidth(TW) product. The TW product is a fundamental parameter that characterizes a given SAR scenario and its corresponding reference function.

The TW product is equal to the “Azimuth Compression Ratio” -the ratio of the width of the physical antenna beam to that of the synthetic beam.

It is also equal to both the number of pulse returns in the synthetic aperture and the number of discrete sample points required in the SAR reference function.

Outline of Presentation

- **Application: Synthetic Aperture Radar (SAR)**

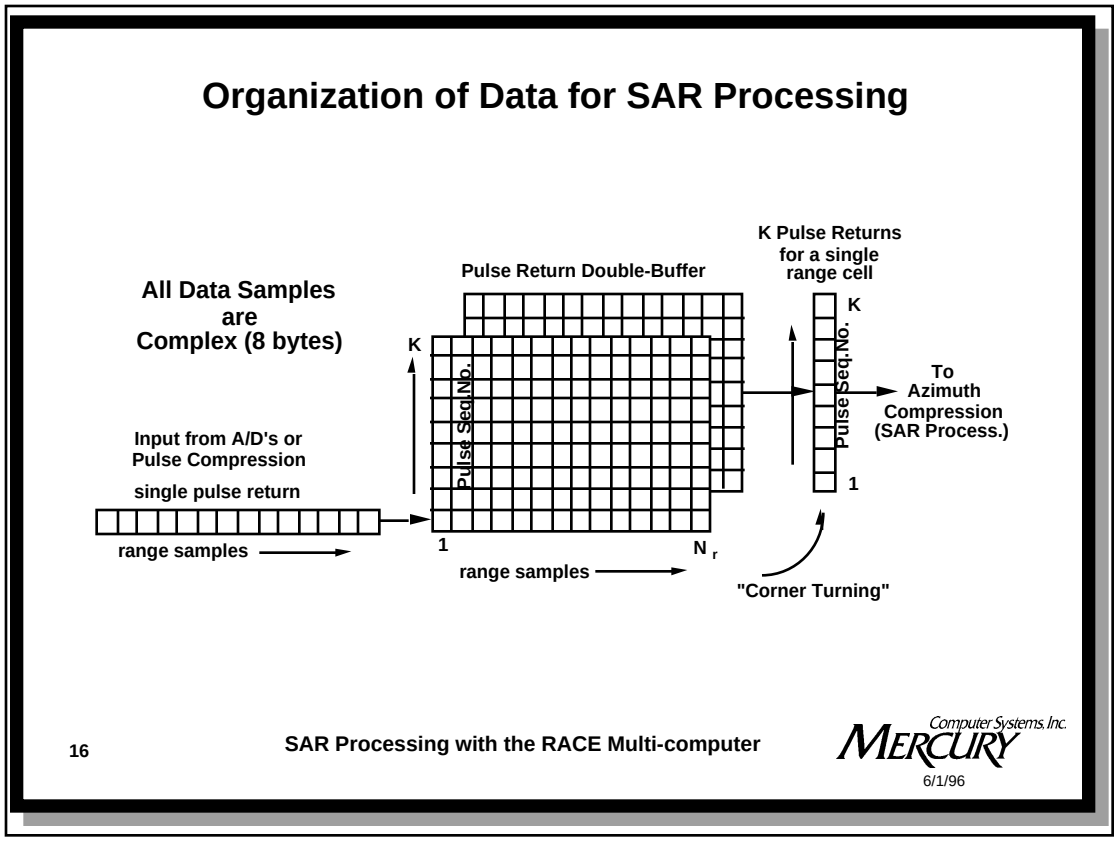
» **Static Description: SAR Signal Processing**

- **Dynamic Description: Real-Time SAR Signal Processing**
- **Resource Requirements: Processing and Memory**
- **Mapping SAR Computations onto *RACE***

SAR Processing

- **Complex Signal Formulation (**analog or digital)
  - preserves the phase information in the signal
- **Range Compression - Matched Filtering / Correlation**
  - individually for each pulse return
- **Corner Turn**
  - re-order data from range-sequential to pulse-sequential (needed when using DRAM memories and/or distributed buffers)
- **Motion Compensation**
  - individually for each pulse return (all range cells)
- **Azimuth Compression - Matched Filtering / Correlation**
  - individually for each range cell, across all pulse returns in the synthetic aperture
- **Image Formation**
  - magnitude the complex outputs of Azimuth Compression

This slide lists the principal operations required for SAR processing, in the order in which they are performed.



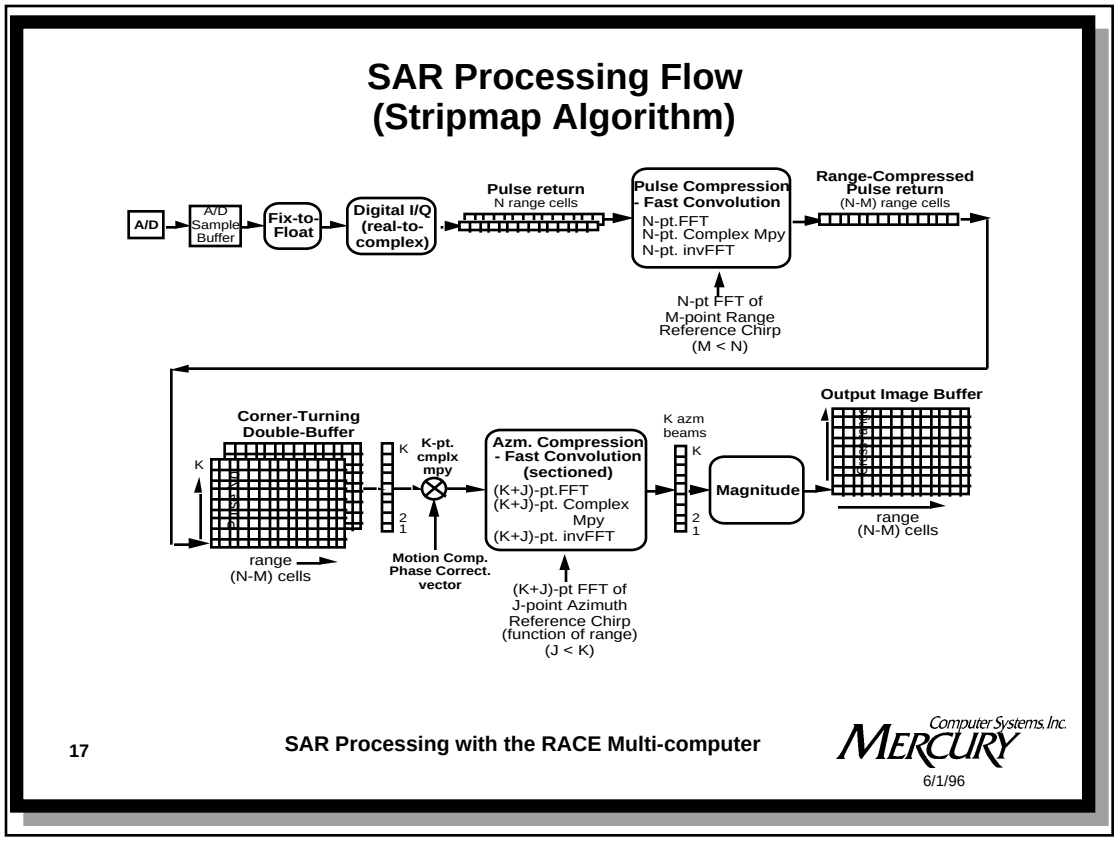
This slide illustrates the organization of pulse return data for SAR processing.

Individual pulse returns are collected by sampling the detected radar echoes with an A/D converter in bursts at sampling rates that typically range from 50 -500 MHz.

These samples (generally 6-12 bits each) are then buffered and read into the signal processor at an average rate of from 5 -50 Msamples/s.

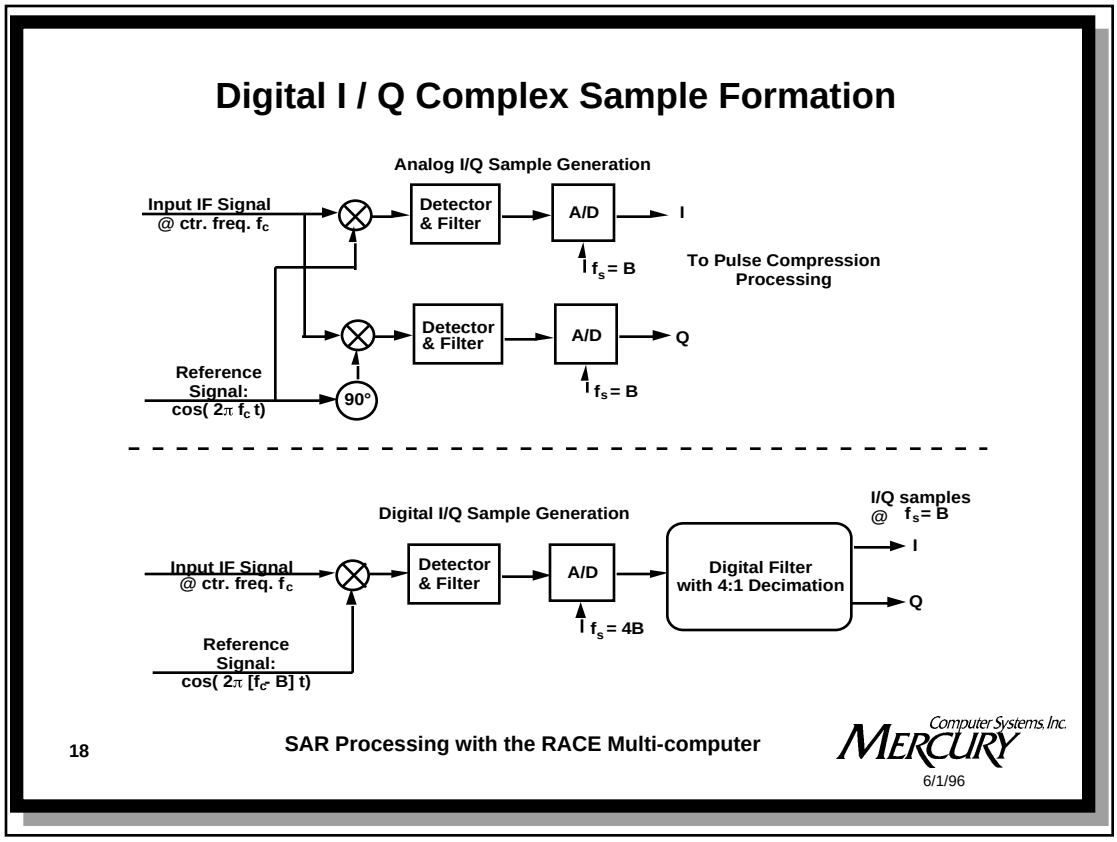
The pulse return samples are read into the rows of a 2-D array in the memory of the signal processor. Thus the rows of this array represent pulse return number, while the columns represen target range.

Pulse compression is performed individually on each received pulse return (i.e. row). The 2-D array of range-compressed samples is then “corner-turned” (i.e. matrix-transposed), and SAR processing (azimuth compression) is performed individually for each range cell (i.e. column).



This slide is a (static) block diagram illustrates the overall SAR processing flow outlined in the preceding slides.

A more detailed description of the processing performed by each of the major blocks in this diagram will be given in the slides to follow.

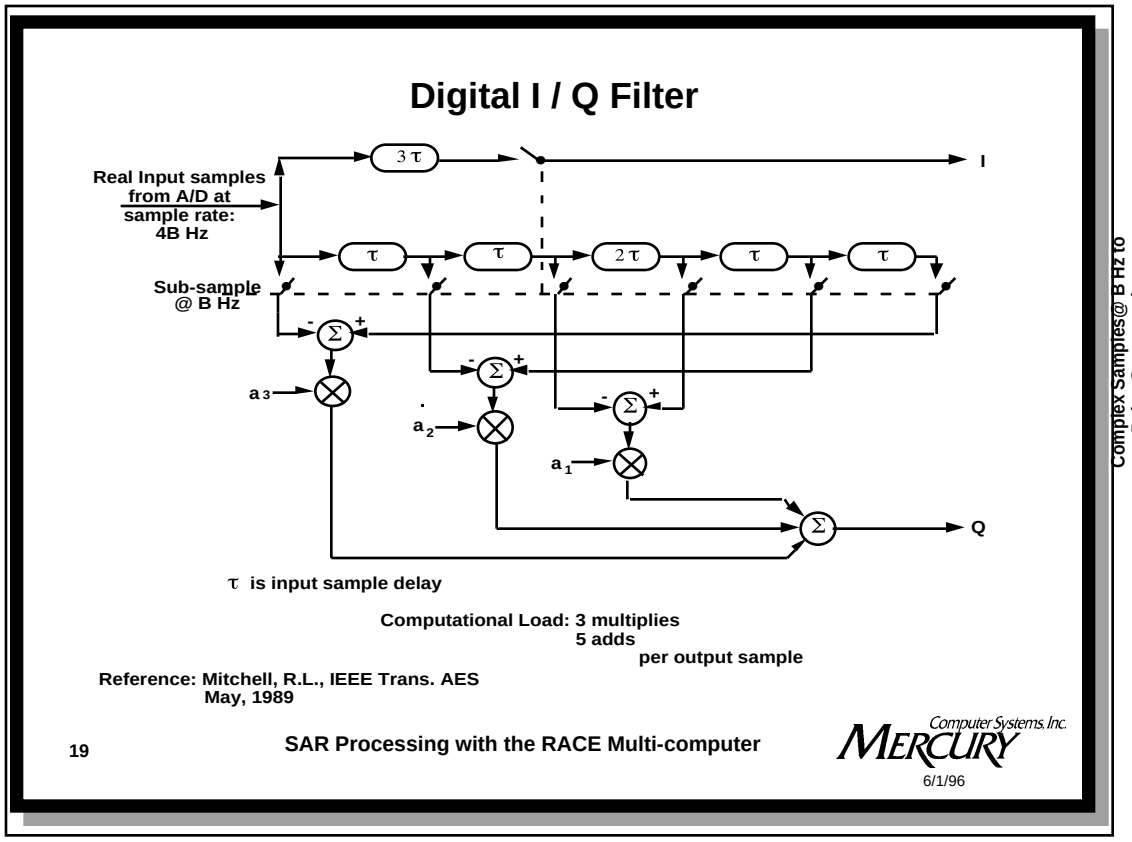


This slide illustrates two alternate methods of complex signal formation.

The top diagram illustrates the “traditional” method in which the I and Q (i.e. real and imaginary) parts of the complex signal samples are formed using analog circuitry and two A/D converters.

The lower diagram shows the more modern approach that uses fewer analog components at the expense of a higher sampling rate and more digital processing. Several variations of this theme are possible. All require sampling at least twice the Nyquist frequency, B. Generally, the lower the sampling frequency, the more processing will be required to form each complex signal sample.

Both approaches illustrated above produce complex signal samples at exactly the same sample rate.



This slide illustrates a particular implementaion of a digital filter for generating complex samples at rate B from real samples at rate 4B.

Pulse Compression

- **Purpose:**

  Simultaneously:

  – **Maximize system sensitivity without using high peak power.**

  » Sensitivity proportional to pulse energy: P ΔT

  » Transmit a long, low-power pulse.

  – **Obtain fine range resolution without using a short pulse.**

  » Linearly sweep the pulse carrier frequency over B Hz.

  » For large B ΔT (> 100), $\delta R \approx c / (2B)$

- **Processing**

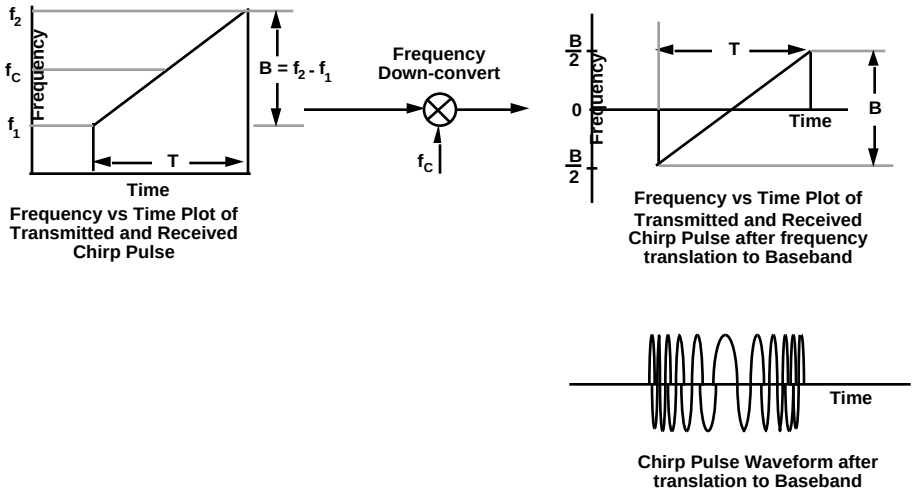
  – **Very similar to that of SAR**

  – **Both based upon correlation processing of linearly-swept frequency signals.**

This slide summarizes the purpose and nature of pulse compression processing.

It is seen to be very similar to stripmap SAR processing. Both pulse compression and stripmap SAR processing are based upon correlation processing of linearly swept-frequency signals.

Pulse Compression Waveform Generation



Frequency vs Time Plot of Transmitted and Received Chirp Pulse

Frequency vs Time Plot of Transmitted and Received Chirp Pulse after frequency translation to Baseband

Chirp Pulse Waveform after translation to Baseband

SAR Processing with the RACE Multi-computer

Computer Systems Inc. MERCURY

6/1/96

This slide reviews the process used to generate the swept frequency transmitted pulse waveform required for pulse compression processing.

Note the similarity of the resulting waveform to that illustrated previously for SAR processing.

The principal differences are:

1. The time/frequency scales are very different for the two cases.

2. The linearly-swept frequency (“chirp”) waveform transmitted by the radar is fixed. That generated by the motion of the radar platform for SAR is variable, being a function of target range and platform velocity.

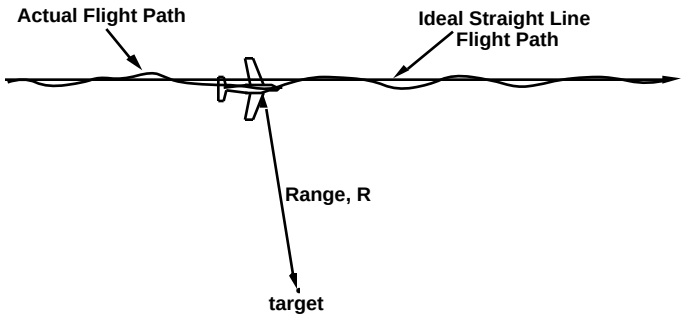
Comparison of Pulse Compression with Azimuth Compression

Attribute	Pulse Compr.	Azimuth Compr.
• **Processing**	Linear Chirp Correlation	Linear Chirp Correlation
• **Signal Duration**	Unc. Pulse Duration, τ (fixed) (2 -100 usec)	$\frac{R\lambda}{2V\delta x}$ (0.1-10 sec)
• **Sample Rate**	f_s = TW / τ (10 -500 MHz)	prf (0.3 -10 KHz)
• **TW Product** (Compression Ratio / No. Ref. Fn. Samples)	$\frac{c\tau}{2\delta x}$ (200 -2K)	$\frac{R\lambda}{2\delta x^2}$ (500 -5K)

This chart summarizes the similarities and differences between the “chirp” waveforms for pulse compression and SAR processing.

It is seen that although the time/frequency scales of the two waveforms are very different, the resulting Time-Bandwidth (TW) products are nearly identical.

Need for Aircraft Motion Compensation



- Doppler Frequency is derived from pulse-pulse phase measurements
- Phase is related to range as follows: $\phi(t) = \frac{4\pi R(t)}{\lambda}$
- Typical values of radar wavelength, λ, are 1 -10 cm
- Allowable phase errors are << 20°. Thus deviation of radar location from assumed straight-line path must be measured (using an INS system) to less than 0.05λ (i.e. 0.5 - 5 mm) and the phase of pulse returns must be corrected correspondingly, before further processing

This slide summarizes the need for platform motion compensation.

This is primarily an issue with regard to airborne SAR, where the aircraft flight path is subject to random disturbances due to atmospheric turbulence and airframe vibration.

It is seen that the deviations that must be compensated for are very small.

Fast Convolution / Correlation

- **Direct Cross-Correlation:**
  - Data Sequence: f(i), i = 1,..,N; with kernel g(i), i =1,...,M < N

$$y(j) = \sum_{i=0}^{N-1} f(i)g^*(i+j); \qquad j = 0,...,(N-M)$$

$$g(i) = 0; \qquad i > M$$

  - Non-Null Computations: M*(N-M+1) complex adds and multiplies

- **Fast Convolution:**
  - Fourier Convolution Theorem: $Y(\omega) = F(\omega) \bullet G^*(\omega)$
  - $F(\omega) = FFT[\ f(i)\ ]$; $G(\omega) = FFT[\ g(i)$ (*zero-padded to N points*)]
  - $y(i) = FFT^{-1}[F(\omega) \bullet G^*(\omega)]$
  - All FFTs are of length $2^k \geq N$ (assume $N = 2^k$)
  - $(N \log_2 N + N)$ complex multiplies; $2\ N \log_2 N$ complex adds
  - Usually faster if $N \geq M$ (length of g(i)), and $M \geq 2 \log_2 N$.

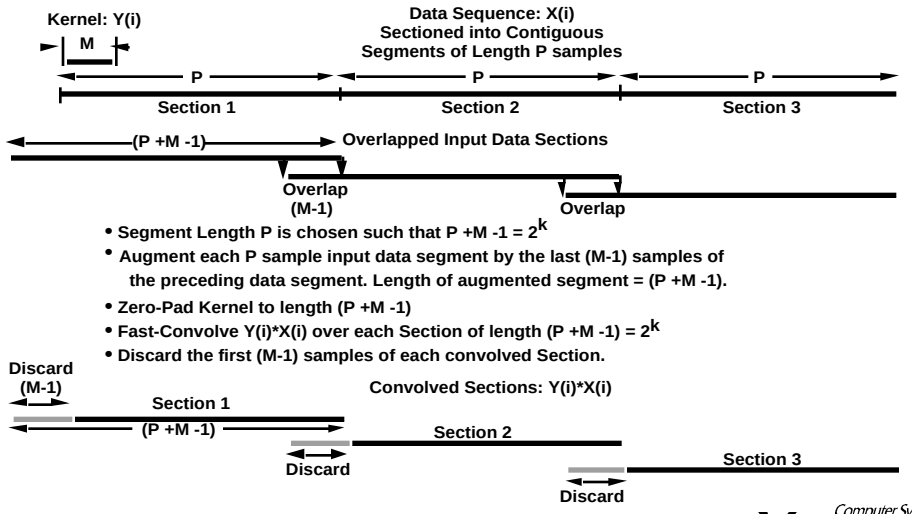
Both pulse compression and azimuth processing consist of correlating the received signals with their corresponding reference waveforms.

It is usually computationally more efficient to implement this correlation process by Fast Convolution, which consists of an FFT, a vector complex multiply, and an Inverse FFT.

This slide describes the Fast Convolution process and its relation to direct (multiply and sum) convolution.

Sectioned Convolutions

- **Used when the length, N, of the data sequence is at least twice as long as that of the kernel (N ≥ 2M) or when latency is an issue.**
  - special case is an unending data sequence



25 SAR Processing with the RACE Multi-computer Computer Systems Inc. MERCURY 6/1/96

Fast Convolution is a batch process -the batch size is that of the FFT's used in the implementation of this process.

How does one apply this batch process to convolve a reference function with a data sequence of indeterminate length -such as the continuous sequence of pulse returns received by a SAR radar?

The answer is to divide the data sequence into overlapped sections and then to perform a Fast Convolution of each overlapped section individually, discarding a part of the part of the output that corresponds to the amount by which the input data sections are overlapped. This process is described in this slide.

The result is exactly identical to that that would be obtained if the convolution were performed directly on the unending input data sequence.

Outline of Presentation

- **Application: Synthetic Aperture Radar (SAR)**
- **Static Description: SAR Signal Processing**
- » **Dynamic Description: Real-Time SAR Signal Processing**
- **Resource Requirements: Processing and Memory**
- **Mapping SAR Computations onto *RACE***

When SAR processing is to be performed in either real-time or fast time, the rate at which the required computations must be performed usually exceeds the processing capability of any given single processor.

Therefore, the only way in which the requisite computations can be completed in real-time is to distribute them over a large number of processors that operate in parallel.

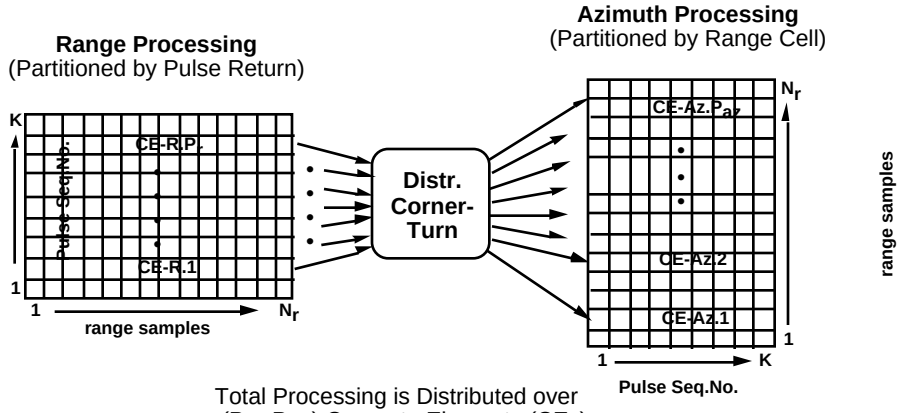
The parallelization of SAR processing and the time-sequence of the computations are described in this Section.

Real-Time SAR Processing

- **The Stripmap SAR Processing Algorithm is necessarily implemented in the following sequentially-executed stages.**
  - Data Conditioning
  - Range Processing
  - Corner Turning
  - Azimuth Processing
  - (some of the above stages may be combined into a single stage)
- **The processing may be pipelined to improve computational throughput**
  - each of the above Stages becomes a stage in the pipeline and is implemented by a different processor, all operating in parallel.
- **A further increase in throughput may be obtained by distributing the computations for any given stage among multiple processors, operating in parallel**

This slide describes the approach to distributing the SAR process over multiple processors so as to achieve increased throughput.

Parallelization of SAR Processing



Total Processing is Distributed over
(P_r +P_{az}) Compute Elements (CEs):

Range Processing over: P_r CEs,
Azimuth Processing over: P_{az} CEs.

SAR Processing with the RACE Multi-computer

Computer Systems Inc. MERCURY

6/1/96

This slide describes an example of process distribution for the range and azimuth compression processes, and illustrates the division of the data among the parallel processors at each stage.

The range processing of different pulse returns may be performed independently for each pulse return. Hence range processing is distributed over multiple processors by pulse return. Each of the parallel Processors in the range processing stage is assigned the processing of a given set of pulse returns.

Similarly, the azimuth processing of different range cells may be performed independently for each range cell. Hence azimuth processing is distributed over multiple processors by range cell. Each of the parallel Processors in the azimuth processing stage is assigned the processing of a contiguous set of range cells.

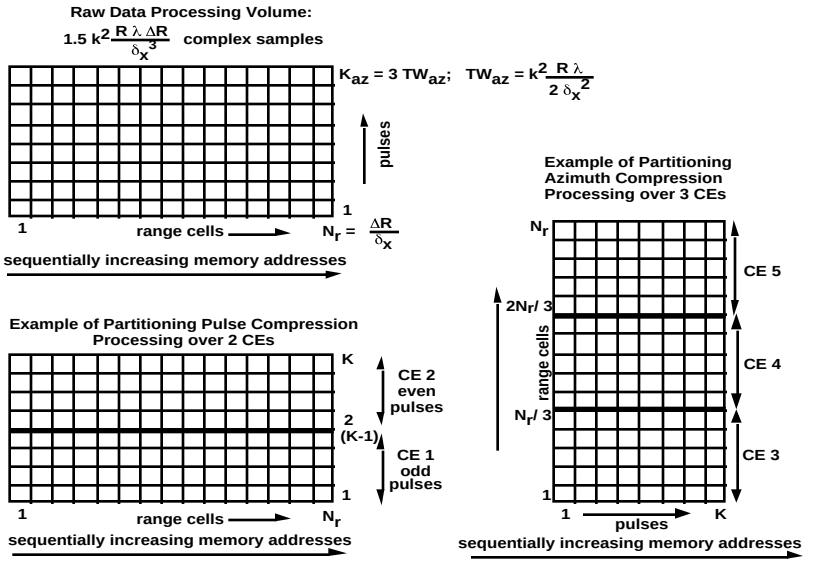
Since range processing is carried out in the range dimension, while azimuth processing is carried out in the azimuth dimension, the data must be explicitly "corner-turned" between these two stages.

This "corner-turn" operation involves two steps:

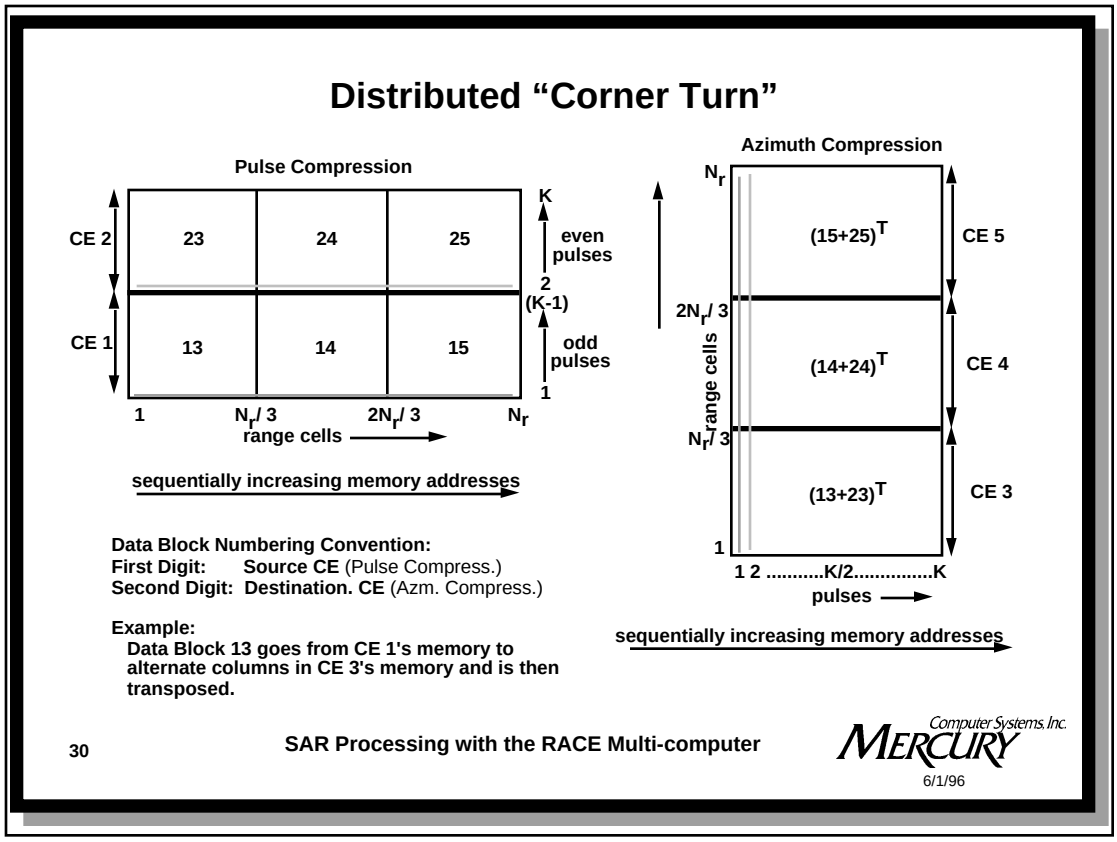
1. The distribution of data from each range CE to all azimuth CEs.

2.The local transposition of the resulting data in each azimuth CE.

Example of Mapping of SAR Processing onto Multiple Processors (2 range and 3 Azimuth)



This slide illustrates a specific example of range processing being distributed across two processors and azimuth processing being distributed across three processors.

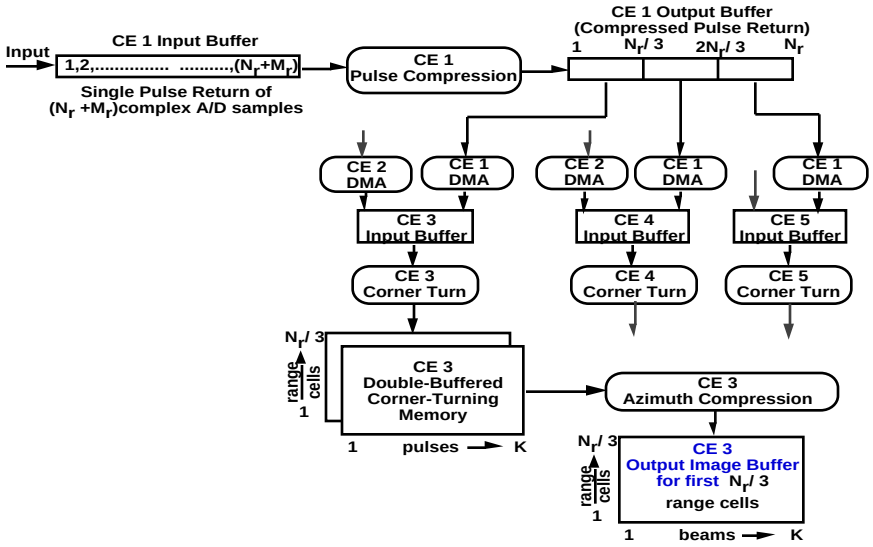


This slide illustrates the distributed “corner-turn” or matrix transposition of data between range and azimuth processors for the specific example illustrated in the preceding slide.

We assume that the even numbered pulses are processed by CE 1 while the odd-numbered pulses are processed by CE 2. This is more efficient (i.e. saves memory) than buffering up the pulse returns into two batches so that the first batch is processed by CE1 while the second batch is processed by CE2.

The first step of the two-step “corner-turn” operation is to distribute each pulse return (i.e. row) from each range CE to all three azimuth CEs. After K pulses have been distributed from the output of the range CEs to the azimuth CEs in this way, the next step is to do a local transpose of the resulting data in each azimuth CE so that consecutive pulse return samples for each range cell will appear in consecutive memory locations in each azimuth CE’s local memory.

Data Distribution from Range Processor: CE 1 to Azimuth Processors: CE 3-5 via Chained DMA



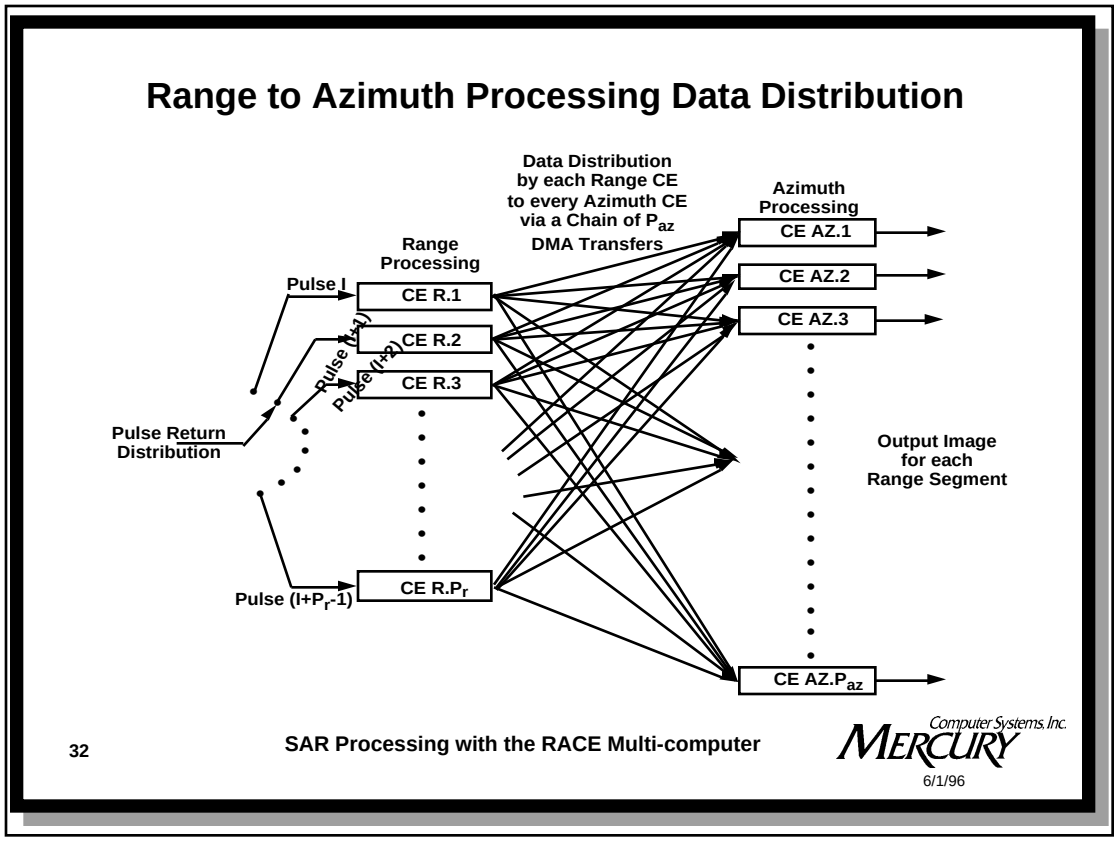
SAR Processing with the RACE Multi-computer

Computer Systems Inc.
MERCURY

6/1/96

This slide illustrates the distribution of data from range processing CE 1 to the three azimuth CEs, for the case of the preceding example, using chained DMA transfers on CE 1.

This distribution is the first step of the two-step distributed corner-turning operation.



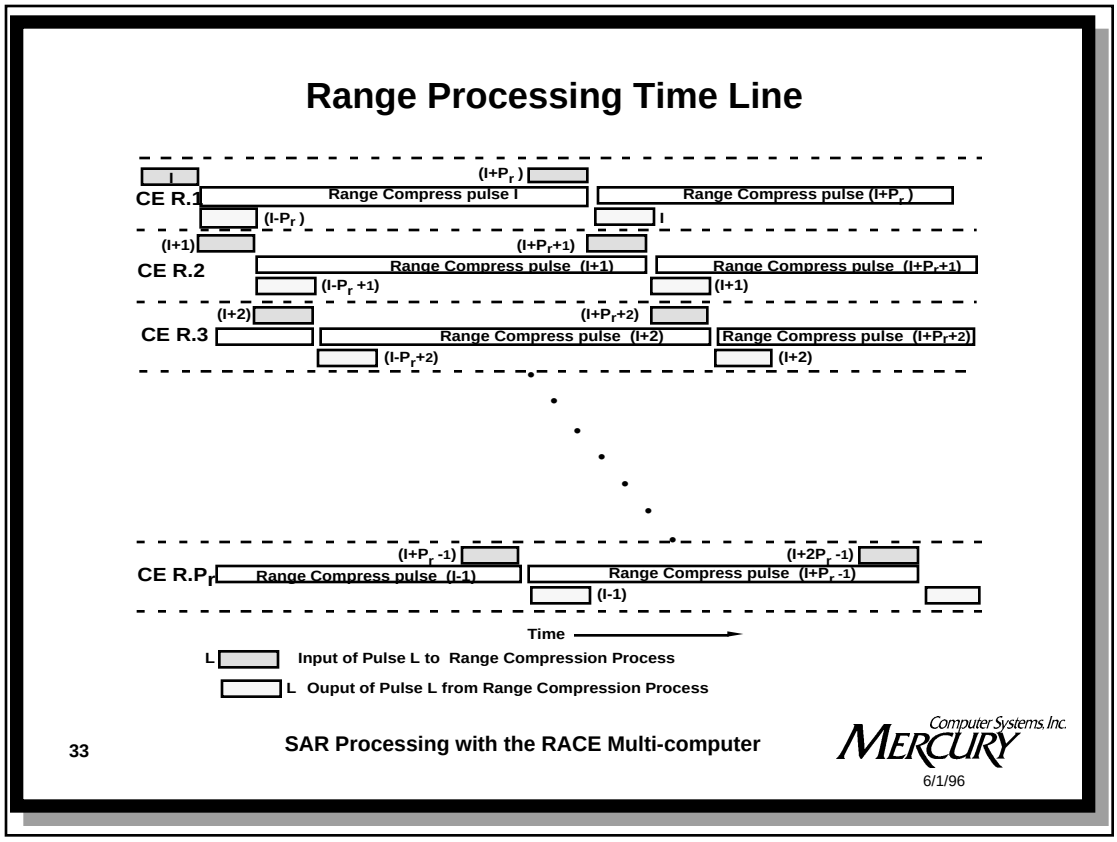
This slide illustrates the flow of data in general case of a SAR processor consisting of P_r range processing CEs and P_{az} azimuth processing CEs.

The incoming pulse returns (each consisting of an N_r-vector of complex range sampes) are distributed to each of the range processing CEs on a round-robin basis.

After each of the P_r range processing CE completes the compression processing of its current pulse, it distributes a different range slice of the resulting compressed pulse return to each of the P_{az} azimuth processing CEs, using chained DMA transfers. This is done by all P_r range CEs in sequence. These inter-processor data transfers constitute the first step of the two-step distributed corner-turning operation.

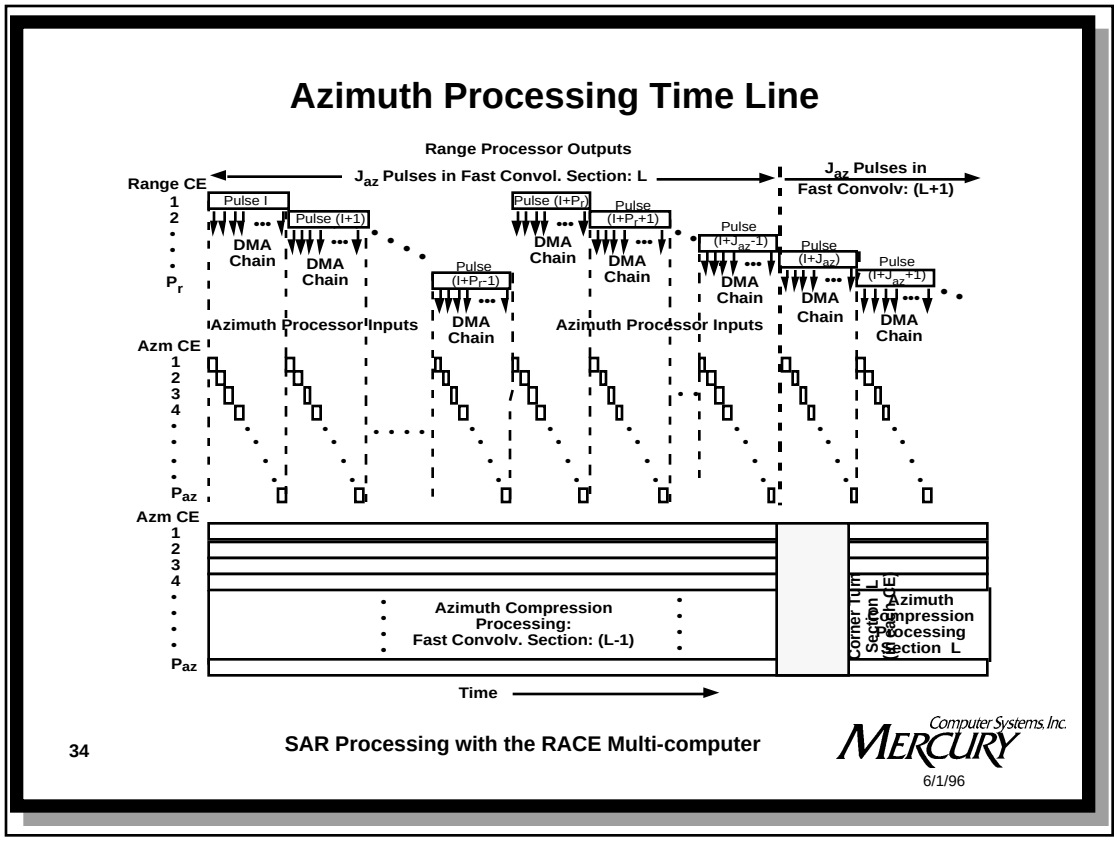
Each of the paths emanating from a given range processing CE to an azimuth CE represents one of the link's in that range processing CE's DMA chain. Every azimuth CE receives data from a specific range slice of each pulse return.

After receiving data from J_{az} consecutive pulse returns (J_{az} is the Fast Convolution Section Length; generally $J_{az} >> P_r$), each azimuth CE does a local transpose of the data in its memory (i.e.step 2 of the two-step distributed corner-turn operation), and begins azimuth processing of its range slice.



This slide illustrates the “round-robin” time sequence of range compression processing for each of the P_r range processors.

In the upper-left-hand corner of this figure, the complex vector of N_r range samples for Pulse Return I is input to range processor CE R.1, and that CE begins the compression processing for that pulse return. While pulse Return I is being processed by CE R.1, the next Pulse Return, (I+1), is input to CE R.2, etc., until Pulse Return (I +P_r -1) is input to and processed by the last range processor: CE R.P_r. The next pulse return in sequence, Pulse Return (I +P_r), is then again input to CE R.1 while that CE completes its processing of Pulse Return I. As soon as CE R.1 completes its processing of Pulse Return I, it outputs its results to the azimuth CEs (by chained DMA) and continues with the processing of Pulse Return (I +P_r), etc.

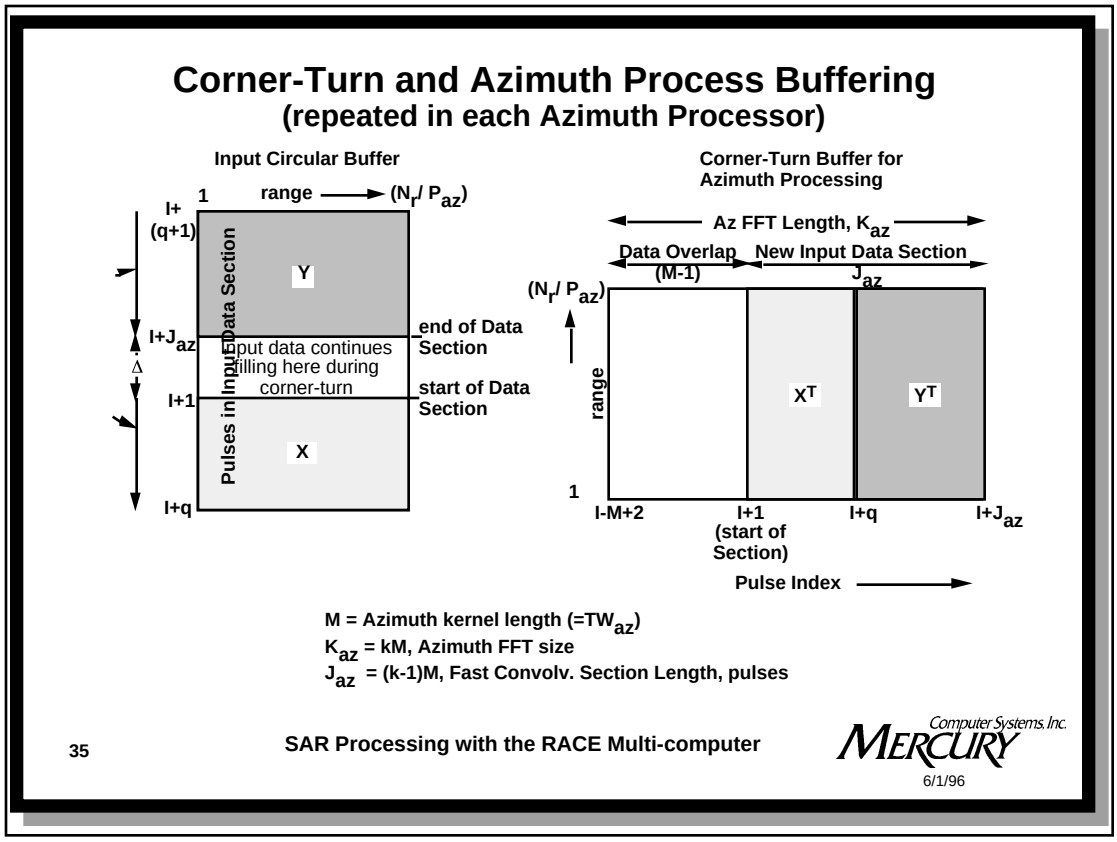


This slide illustrates the distribution of the range-compressed pulse returns from each of the P_r range processing CEs to <u>all</u> P_{az} azimuth processing CEs by chained DMA transfers, and the concurrent azimuth compression processing by each of the azimuth processing CEs.

As discussed in the preceding slide, each range processing CE dsitributes its range-compressed output vector, consisting of N_r complex samples, to all P_{az} azimuth processing CEs by a sequence of chained DMA transfers, for every pulse return processed. Each azimuth processing CE therefore receives a pulse return range segment consisting of approximately (N_r/P_{az}) complex samples.

This continues until one Azmuth Fast Convolution Section, consisting of J_{az} pulse returns, has been range-compressed and accumulated in the input buffer of each azimuth processor. Each azimuth processor then performs a local corner-turn of the data in these J_{az} pulse returns and then begins the azimuth compression process, individually for each of the (N_r/P_{az}) range samples in its range segment.

The azimuth processor input buffering scheme that allows the pulse returns for the next Azmuth Fast Convolution Section to be accumulated while the data from the previous section is being corner-turned is illustrated in the next slide.



This slide illustrates the input buffering scheme used by each of the P_{az} azimuth processors. The buffer consists of two sections. The first section is an input circular buffer of size: $(J_{az}+\Delta)$ x (N_r/P_{az}) complex samples, where J_{az} is the azimuth section length, in pulses, and Δ is the a value somehat greater than the number of pulse returns that will be input while the corner-turning process is being performed.

An azimuth fast convolution data section can begin and end at any row in this circular buffer. When the end of the buffer is reached, the remaining data in the current data section "wraps-around" and the next pulse return in the section is written into the first row of the buffer. When an azimuth data section of J_{az} pulses has been accumulated, that section is corner-turned from the input circular buffer into the "working" buffer section of size: K_{az} x (N_r/P_{az}) complex samples, where K_{az} is the azimuth FFT size; $K_{az}=(J_{az}+TW_{az})$. Note that the "corner-turning" operation cannot be done "in-place" and requires simultaneous access to both buffers.

Azimuth processing is then performed on all the data in the "working" buffer section (i.e. the new data section plus the overlapped pulses from the preceding section) while new range-compressed pulse return data continues to be written into the input circular buffer section. The fact that the length of the input circular buffer is somewhat greater (i.e. by Δ rows) than J_{az} pulses allows new data to continue to be written to this buffer while corner-turning of the remaining data in the buffer is in progress.

Outline of Presentation

- **Application: Synthetic Aperture Radar (SAR)**
- **Static Description: SAR Signal Processing**
- **Dynamic Description: SAR Signal Processing**
- » **Resource Requirements: Processing and Memory**
- **Mapping SAR Computations onto *RACE***

Resource requirements will be analyzed in two parts: Processing Power requirements and Memory requirements.

Computational Loading -Approach

- **Total Computational Load**
  - Computational Load per Input Sample x Data Rate (samples / sec)

    (flop / sample) x (Msamples / sec) = Mflops

- **Computational Load / sample**
  - Sectioned Convolution
    - » Range Compression
    - » Corner Turn (req'd when using DRAM Memories and/or distributed buffers)
    - » Azimuth Compression
  - Total: (Range + Azimuth Compression)

- **Data Rate**
  - (Number of range samples / pulse) x (pulses / sec) = samples /sec

    Date Rate: $Q = N_r$ x prf samples / sec

The processing power required for a given SAR application is determined by the total computational load in Mflops. The total computational load can be simply expressed by determining the computational load per input data sample (in flops/ sample) and then multiplying this number by the input data rate in Msamples/s. The result will be expressed in Mflops.

Note that the computational load per input data sample is an intrinsic attribute of the computational process to be performed, while the data rate is not. The average, sustained data rate, in samples/sec, is merely the product of the number of range samples/pulse return times the pulse repetition frequency in Hz.

The total computational load per sample is found by adding the per-sample computational load for all stages in the SAR process. This load will be dominated by that of the sectioned, fast-convolutions that are used for both the range and azimuth compression processes. Hence the computational load per sample of a sectioned fast convolution will be determined first.

Finally, it should be noted that the determination of the total computational load in Mflops, as described above, is not the whole story with regard to determining the number of CEs required because not all Mflops are created equal - a given CE performs some operations more efficiently than others. This issue will be addressed later.

Computational Load - Sectioned Convolution

- **FFT Butterfly (BFY)**
  - 1 complex multiply, 2 complex adds = 10 real op'ns
- **Fast Convolution**

– FFT	$N / 2 \log_2 N$ BFYs
– N-pnt cmplx mpy	N cmplx mpy's = 6 N real op'ns
– IFFT	$N / 2 \log_2 N$ BFYs
– Total	$(10 N \log_2 N + 6 N)$ real op'ns

This slide illustrates the number of real operations required to perform an N-point fast convolution. A radix 2 FFT is assumed for simplicity. The use of a larger radix FFT would reduce the above results by about 10%.

Performance / Memory Trade-offs for Sectioned Fast Convolutions

- **N-point Fast Convolution:**

  $S = 10\ N \log_2 N + 6\ N$ real operations

  - Sectioning the Convolution:

  Sub-divide the Data Sequence into Sections of Length P such that: $P \geq 2M$, where M is the length of the Convolution Kernel

  Sectional FFT of Length: $N = P + M - 1 \approx P + M$

  - An N (= P + M -1) point FFT is done for every Data Section of length P
  The FFT size (N) determines the amount of buffer memory required.

  - Computational Load per Data Sample: X = S / P

  $$X = 10\ \frac{(P+M-1)}{P}\ \log_2(P+M-1)\ + 6\ \frac{(P+M-1)}{P}$$

  Define: $k = \frac{N}{M}$; Then: $P \approx (k-1)\,M$

  $$X = 10\ \frac{k}{(k-1)}\ [\ \log_2 k + \log_2 M + 0.6\ ]$$ real op'ns / data sample

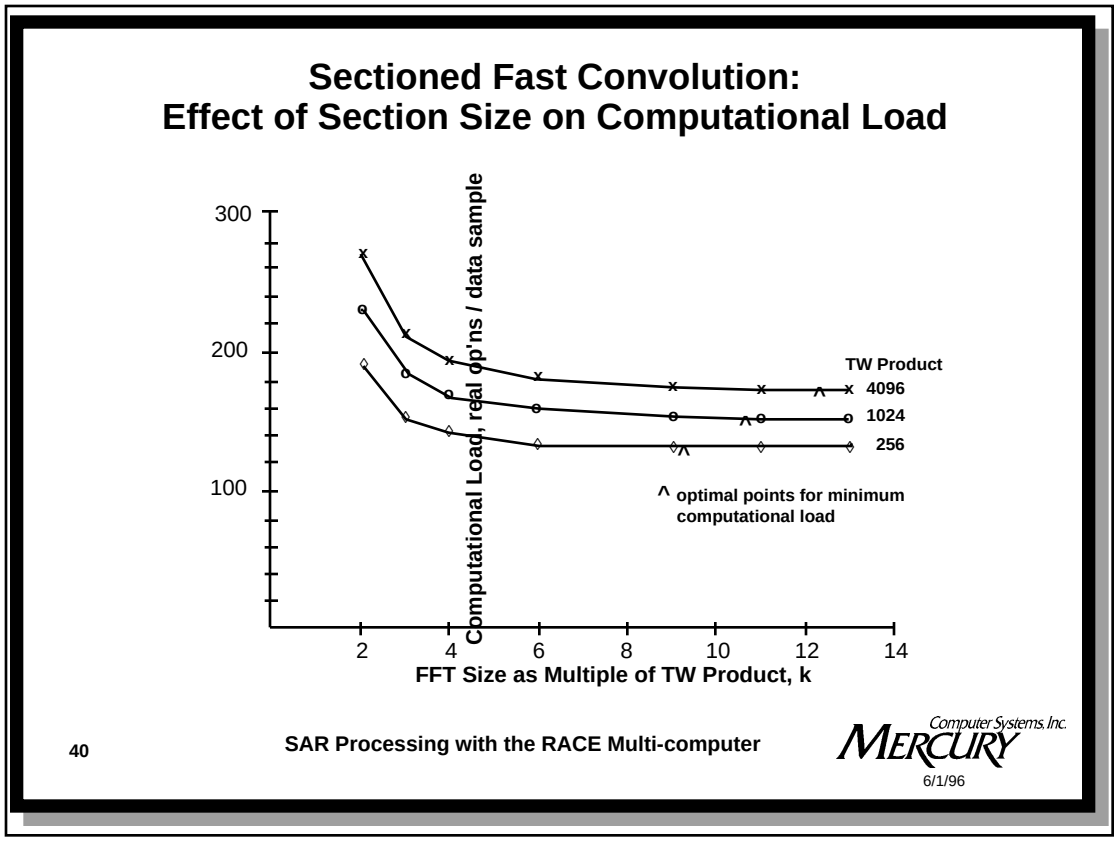
- **Computational Load, X, is minimized at an optimum value of k.**

- **Buffer Memory Requirements, however, increase in proportion to k.**

This slide illustrates the computation of the number of real operations required per sample for a sectioned fast-convolution. The result is a function of the kernel length (i.e. TW product), M, and the FFT size, N.

For simplicity, the FFT size is expressed as a multiple of the kernel length: i.e. N = kM. It turns out that there is an optimal FFT size that minimizes the number of real operations per sample. On the one hand increasing the FFT size reduces the percentage overlap between adjacent data sections thereby decreasing the effective number of operations required per input sample. However, increasing the FFT size also increases the number of operations per sample by $\log_2 N$. An optimum point is reached where these two conflicting trends counter-balance each other.

However, minimizing the total computational load by operating at the optimum value of FFT size must also be balanced against the fact that the amount of buffer memory required increases in direct proportion to the size of the FFT.



This slide shows the variation of the per-sample computational load for a fast convolution with FFT size, for three different values of kernel size (i.e. TW product). Note that depending upon the kernel size, the FFT size that minimizes the per-unit computational load is on the order of ten times the kernel size.

However, since the amount of memory required increases in direct proportion to the FFT size, and since the minima are very flat, operating at these minimum points is often not cost effective. In cases where the amount of memory required is determined solely by the FFT size, it is usually much more cost-effective to endure a slight reduction in processing efficiency (and thus increase the total processing requirement) in order to achieve a three-to-five fold reduction in memory requirement.

Trade-off Increased Processing for Greatly Reduced Azimuth Compression Memory Req'ts

- **Azimuth Compression Buffer Memory Req'ts drive Total System Memory Req'ts.**
  - directly proportional to: Azimuth FFT Size, Number of Pulses in Section Length.
- **Choosing Azimuth Data Section Length to Minimize Computational Load of Azimuth Compression results in excessive Azimuth Processor buffer memory requirement.**
  - Optimum FFT size ≈ 10 TW_{az} (depends on TW_{az})
- **Memory Requirements can be reduced by allowing computational load to increase. -Tradeoff increased computation for less memory**
  - Curves of computational load vs memory are very flat near the optimum (minimum computation) point.
  - Choosing k =2 results in an 80% reduction in FFT size (and memory req't) at a cost of ≈50% increase in computational load.

This slide discusses the trade-offs between minimizing processing load and minimizing memory requirements for fast convolutions.

This trade-off generally applies only to azimuth compression processing, where the amount of buffer memory required is proportional to the azimuth FFT size.

In the case of range compression processing, the amount of memory required for buffering each pulse return is fixed by the number of range samples collected per pulse, rather than by the range FFT size. Hence the preceding trade-off argument does not apply. For range compression processing, an FFT size should be chosen that minimizes the total processing requirement, including edge effects.

Pulse Compression Processing Load

- **Assume Sectioned Convolutions with Section Length P ≈ 10 M_r.**
  - M_r is number of samples in range correlation kernel
    (M_r = TW (BT) product of Chirp pulse)

- **All processing load values are number of <u>real operations per complex sample</u> of input data.**
  - **Fix-to-Float:** **7**
  - **Digital I / Q** (real-to-complex)**:** **8**
  - **Fast Convolution (complex):** **$11\log_2 M_r + 45$**

- **Total** (real op'ns / complex sample) **$11 \log_2 M_r + 60$**

This slide shows the evaluation of the computational load per sample for range processing. It assumes that the only operations required are fix-to-float conversion, digital I/Q complex signal formation and range compression. The computational loading for range compression assumes an FFT size equal to 11 times the kernel size for minimum fast convolution processing load.

Azimuth Compression Processing Load

- **Assume Sectioned Convolutions with Section Length P ≈ 2 M_{az}.**
  - M_{az} is number of samples in azimuth correlation kernel (M_{az} = TW (BT) product of SAR reference function)
  - P = 2 M_{az} -> k =3: (33% section overlap)
- **All load values are number of real operations <u>per complex sample</u> of input data.**

– Corner Turn (complex transpose):	10 (eqv. est.)
– Motion Compensation (complex mpy)	6
– Fast Convolution (complex):	20 $\log_2 M_{az}$ + 32
– Magnitude (squared):	10
• Total	**20 $\log_2 M_{az}$ + 58**

This slide shows the evaluation of the computational load per sample for azimuth processing. The “corner-turn” time represents the local “corner-turning” that has to be performed by each azimuth processor and is based upon actual measurements for complex data.

The computational load for azimuth fast convolution is based upon an FFT length equal to three times the kernel length (33% overlap).

Increasing the overlap to 50% (FFT size = 2 x kernel length) would reduce the memory requirement by 1/3, but would increase the processing requirement by about 25%.

Total Computational Load, Mflops

- **Total Computational Load**
  - Load per unit complex input sample X Sample Rate
  - Sample Rate Q = No. range cells per pulse X prf:
    Q = N_r x prf
- **$N_r = (\Delta R / \delta_x)$**
- **Pulse Repetition Frequency (prf)**

$$prf \geq \frac{2V\theta_B}{\lambda}; \qquad \delta_x = \left(\frac{k\lambda}{2R\theta_B}\right)R = \frac{k\lambda}{2\theta_B}$$; k = No. of indep. "looks"

$$prf \geq \frac{kV}{\delta_x}$$

Sample Rate: $Q \geq \frac{kV\Delta R}{\delta_x^2}$ = k $\frac{\text{V } \Delta\text{R} = \text{ coverage rate; m}^2\text{ / sec}}{\delta_x^2 = \text{ pixel area; m}^2}$

Sample Rate = k times No. of pixels / sec

This slide expresses the average data sample rate Q as a function of the platform velocity: V, the range swath: ΔR, the pixel resolution: δ_x, and number of independent "looks": k.

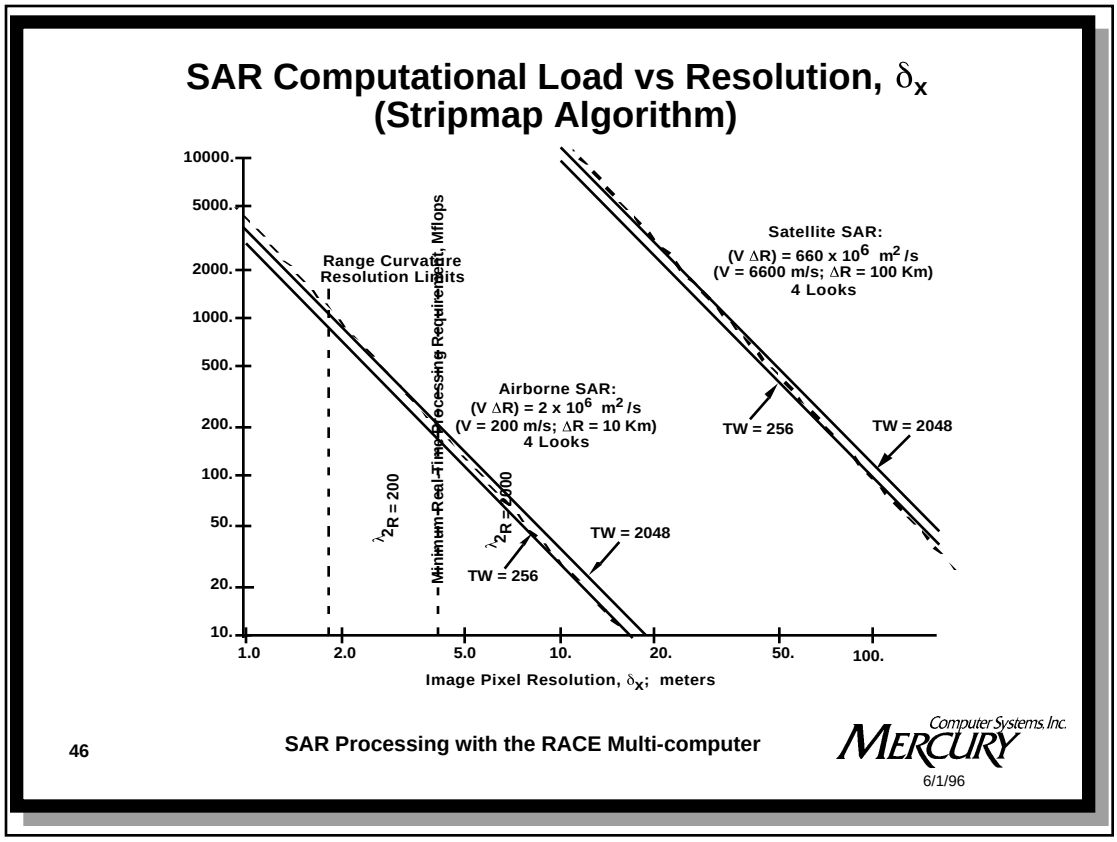
Computational Load, Mflops

- **Total Computational Load**
  - Load per unit complex input sample X Sample Rate
- **Per Sample Computational Load**
  - $T = 11 \log_2 M_r + 20 \log_2 M_{az} + 118$ real operations / sample
- **Sample Rate**
  - $Q = k \frac{V \Delta R}{\delta_x^2}$ complex samples / sec
    (k = no. of independent "looks")

- **Total Load:**
  - $R = k \ [11 \log_2 M_r + 20 \log_2 M_{az} + 118] \ \frac{V \Delta R}{\delta_x^2}$ real opn's/sec

This slide shows the evaluation of the total processing requirement as a function of area coverage rate in pixels/sec, the range and azimuth compression TW products, and the number of "looks"

The total processing requirement per sample is obtained by adding the results of slides 42 and 43. This total is then multiplied by the average data rate, expressed as a function of platform velocity, range swath and resolution, from slide 44.



This slide is a plot of the expression for total real-time computational load as a function of resolution that was developed in the preceding slide. Two distinct cases are illustrated, a typical airborne SAR and a typical satellite-borne SAR. For purposes of illustration the range and azimuth compression time-bandwidth products are assumed to be identical

For constant time-bandwidth product, the processing load is seen to increase inversely with the square of the image resolution. However both the range and the azimuth compression time-bandwidth products will also generally increase in inverse proportion to the image resolution, and the processing load increases as the logarithm of these TW products. Hence the total processing load will increase somewhat more rapidly than the inverse-square of the resolution. However, it is seen that for time-bandwidth products greater than 256, the effect of increasing time-bandwidth product on processing load is relatively small.

Also illustrated are the maximum $\lambda^2 R$ range curvature limits on the minimum resolution that may be obtained using the Stripmap processing algorithm. Stripmap processing is only valid at resolutions to the right of the corresponding range curvature limit for a given value of $\lambda^2 R$. In order to obtain resolutions finer than that indicated by a given $\lambda^2 R$ limit, another algorithm such as range-migration processing must be used.

SAR Process Memory Requirements (1 of 2)

- **Pulse Compression Processing**
  - Each Pulse Comp. CE requires both an input and output double-buffer, each of length 2 x N_r complex samples.
  - Total: (2 x 2) N_r samples per processor, or **$4 P_r N_r$ cmplx samples.** ($N_r \approx$ 1,000 -20,000)
- **Azimuth Compression Processing**
  - Each Azimuth Comp. CE requires at least the following:
    - » Corner-Turning Double-Buffer: 2 x (N_r / P_{az}) x K_{az} cmplx samps
    - » Output Image Buffer: (N_r / P_{az}) x K_{az} real samples
  - Total over all P_{az} CEs: **$2.5 N_r K_{az}$ complex samples** ($K_{az} \approx$ 4,000 -40,000)
- **Total Memory Requirement: ($2.5 N_r K_{az} + 4 P_r N_r$) cmplx samples**
  - Since $K_{az} >> P_r$, and: $N_r = \Delta R/\delta_x$; $K_{az} = 2 TW_{az} = 2 k^2 R \lambda/(2\delta_x^2)$
  - Total Requirement **$\approx 2.5 N_r K_{az} = 2.5 k^2 (R \lambda \Delta R) / (\delta_x^3)$ cmplx samples**

This slide illustrates the evaluation of the total system memory requirement. These requirements are evaluated separately for both the range and the azimuth processors

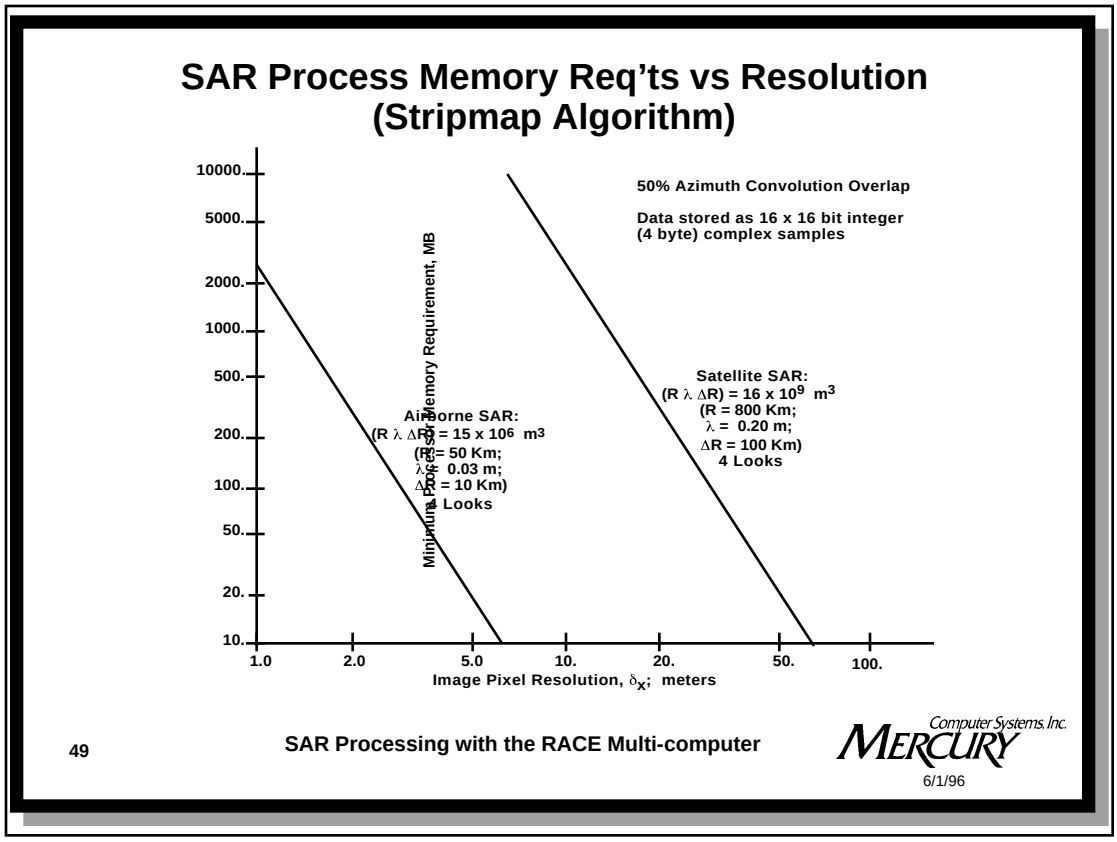
The data memory requirements for the range compression processors are seen to be quite modest -less than 1MB per processor.

Total memory requirements are seen to be dominated by those of azimuth processing -primarily because the size of the corner-turning double buffer that is required. See slide 35. The azimuth processor memory requirement is seen to vary directly in proportion to the product: ($R \lambda \Delta R$) and inversely with the cube of the resolution.

In certain cases it may be permissible to reduce the above memory requirement by up to 50% by storing the complex data in the corning-turning double buffer in a 32-bit fixed-point complex (i.e. 16 I, 16 Q) representation rather than 64-bit complex floating point. All computations would still be performed in floating-point. The data would merely be converted to fixed-point for storage in the corner-turning double buffer after range compression and then converted back to floating point for azimuth compression. However, the feasability of such an approach would have to be evaluated individually for each situation.

SAR Processor Memory Requirements (2 of 2)

- **SAR Processor memory requirements:**
  - dominated by those of the azimuth processors.
  - increase as the inverse cube of the resolution.
- **Requirements can become exceedingly large at fine resolutions.**
- **Need to conserve memory -store complex data as 4-byte samples**
  - Convert intermediate results (complex data) to 16 x 16 bit integers for storage in corner-turning buffers, using block floating point
  - Storing complex numbers as 16 x 16 bit integers provides a 50% memory saving over 32 x 32 bit complex floating point.
  - 16-bit integers provide a 93 dB dynamic range, which is more than adequate, especially for the intermediate results.
  - Re-convert corner-turned data back to 32 x 32 bit complex floating point for azimuth processing computations
- **Total Memory requirement with data stored as 4-byte cmplex samples:**
  - $\approx 10\, N_r\, K_{az} = 10\, k^2 (R\, \lambda\, \Delta R) / (\delta_x^3)$ **bytes**



This slide is a plot of the approximate expression for total SAR processing memory requirement as a function of the resolution, δ_x, that was developed in the preceding slide. Again, two distinct cases are illustrated, a typical airborne SAR and a typical satellite-borne SAR.

The total system memory requirement is effectively concentrated over the azimuth processors and is seen to vary as the inverse cube of the desired image resolution.

Since the processing power required for real-time SAR processing was seen to increase only as the inverse square of the resolution, it can be seen that the memory required per Mflop of azimuth processing power (i.e. MB /Mflop) will increase inversely with resolution.

Summary of Processing Requirements

- **SAR Processing Load**
  - Primarily proportional to the input sample data rate
    - » varies approximately as the inverse square of the desired image pixel resol'n
  - Typical requirements range from 10 -10,000 Mflops (real-time)
- **SAR Processing Memory Requirements**
  - Inversely proportional to the **cube** of the desired image pixel resolution and the (R λ ΔR) product of the Application.
  - Typical requirements range from 10 -20,000 MB
- **Image Resolution is the principal driver of processing and memory requirements.**
  - Memory requirements (MB) increase faster than processing requirements (Mflops) as resolution becomes finer
- **Processing is easily Parallelizable**
  - Data is distributed by pulse return for range processing
  - Data is distributed by range cell for azimuth processing

This slide summarizes the processing and memory requirements for real-time SAR processing. The processing was shown to be easily distributable over multiple processors for parallel computation.

The principal driver of both processing and memory requirements is the pixel resolution of the desired image

Outline of Presentation

- **Application: Synthetic Aperture Radar (SAR)**
- **Static Description: SAR Signal Processing**
- **Dynamic Description: SAR Signal Processing**
- **Resource Requirements: Processing and Memory**
- » **Mapping SAR Computations onto *RACE***

Issues in Sizing a RACE Multi-computer for SAR Stripmap Processing

- **Processing Load**
  - determines total number of CEs required.
  - total Mflop requirement provides only a first cut.
    - » not all Mflops are created equal!
    - » effective Mflop throughput of a given CE must be determined individually for each processing step (type of computation).
- **Memory Requirements** (assume data stored as [16 x 16]-bit complex)
  - determines memory required for each CE. May also determine total number of CEs required.
  - azimuth processors require more memory than range processors.
    - » Range Processor Memory: **$16\ N_r$ bytes** per range processor
    - » Azimuth Processor Memory: **($20\ N_r\ TW_{az}$) /P_{az} bytes** / azm. processor
- **Maximum Bandwidth Requirements**
  - Equal to (2 x Input Data Rate) / (No. of parallel processors/stage)
  - Input Data Rate = **$8\ N_r$ prf = 8 k (V ΔR) /(δ_x^2) bytes/sec**

This slide describes the three principal application attributes that define the multi-computer requirements:

1. Required computational throughput in Mflops
2. Required processor memory for data storage in MB
3. Required inter-processor data communication bandwidth, MB/s

Processing Load and Number of CEs Required

- **1. Determine equivalent Mflop throughput of the selected CE type for each processing step (e.g. fast convolution)**
  - Not all Mflops are created equal!
- **2. Determine total Mflop requirement of each processing step.**
  - Determine number of operations per input sample, for each step.
  - Multiply by input data rate: **$Q = k\,(V\,\Delta R)\,/(\delta_x^2)$** samples/sec.
- **3. Divide 2. by 1. to obtain equivalent number of processors required for each processing step.**
- **4. Sum results of 3. over all processing steps to obtain total number of processors required.**

This slide describes the four steps used to determine the number of processors required to provide the specified computational throughput.

1. Equivalent Mflop Throughput for SHARC CE

• Fix-to-Float /Float-to-Fix Conversion	80 Mflop
• FIR Filter	45 Mflop
• Complex Multiply	27 Mflop
• Fast Convolution	94 Mflop
• Corner-Turn	80 Mflop
• Magnituding (squared)	38 Mflop

This slide describes the effective processor throughput of a SHARC CE for the various processing operations used in Stripmap SAR processing.

Processing Requirements

- **Operations per Sample**
  - Fix-to-Float / Float-to-Fix Conversion: 7
  - FIR Filter for Complex Signal Formation: 8
  - Complex Multiply for Motion Compensation: 6
  - Fast Convolution: (50% overlap) $20 \log_2 TW + 32$
    (10% overlap) $11 \log_2 TW + 45$
  - Corner Turn**:** 10
  - Magnituding: 3
- **Sample Rate**
  - **$Q = k (V \Delta R) / (\delta_x^2)$** samples/sec
    - » k Number of "Looks"
    - » V platform velocity. m/s
    - » ΔR range swath width, m
    - » δ_x resolution, m

This slide describes the processing requirements for the various SAR processing steps that were referred to in the preceding slide. These processing requirements are expressed in terms of the number of real operations per sample and the data sample rate. The actual processing requirement for each step, in Mflops, is equal to the product of these two quantities.

Thus this slide gives the actual processing requirement for each step, whereas the previous slide gave the effective processor throughput for that step, both expressed in Mflops. The effective number of processors required for each step is then found as the processing step requirement divided by the corresponding throughput value from the last slide.

Example of Processor Sizing (Parameter Definitions)

- **SAR System Parameter Values**
  - Resolution, m δ_x = 1
  - Swath Width, m ΔR = 20,000
  - Platform Velocity, m/s V = 200
  - Maximum Range, m R = 100,000
  - Radar Wavelength, m λ = 0.03
  - No. of Looks, k = 1
- **Derived Parameters**
  - Data Rate, samps/s $Q = k(V\,\Delta R)/\delta_x^2 = 4 \times 10^6$
  - Azimuth Proc. Memory, bytes $10\,(R\,\lambda\,\Delta R)/\delta_x^3 = 600 \times 10^6$ $\{20\,N_r\,TW_{az}\}$
  - Range Proc. Memory, bytes $16\,P_r\,(\Delta R/\delta_x) = 320\,P_r \times 10^6$
  - Number of Range Samples $N_r = (\Delta R/\delta_x) = 20 \times 10^3$
  - Reduced prf, Hz $prf = k\,(V/\delta_x) = 200$
  - Azimuth TW product $TW_{az} = (R\,\lambda)/2\delta_x^2 = 1500$

This slide specifies the radar parameters that determine the processor requirements for a typical SAR application. The top half of the slide lists the actual radar parameter values that are needed to specify the SAR processor.

The bottom half of the slide lists the corresponding derived parameters (that are computed directly from the radar parameters) such as data sample rate, memory requirements, azimuth TW product, etc. that define the processor throughput in Mflops, the memory in MB, and I/O bandwidth in MB/s.

Processor Requirement Determination

Processing Step	CE Throughput Mflop	Requirement, Mflop (flops/sample x samps/sec)	No. of CEs Required
Range Processing			
Fix-to-Float Conversion	80	7 x (4×10^6) = 28	0.35
FIR Filter for Digital I/Q	45	8 x (4×10^6) = 32	0.71
Fast Convolution (10% o'lap)	94	166 x (4×10^6) = 664	7.06
Float-to-Fix Conversion	80	7 x (4×10^6) = 28	0.35
Sum		**752**	**8.47**
+ 30% Margin			**2.54**
Total No. of Range Processors:			**11**
Azimuth Processing			
Fix-to-Float Conversion	80	7 x (4×10^6) = 28	0.35
Corner-Turn	80	10 x (4×10^6) = 40	0.50
Complex Multiply	27	6 x (4×10^6) = 24	0.89
Fast Convolution (50% o'lap)	94	252 x (4×10^6) =1008	10.72
Magnitude	26	10 x (4×10^6) = 40	1.54
Float-to-Fix Conversion	80	7 x (4×10^6) = 28	0.35
Sum		**1168**	**14.35**
+ 30% Margin			**4.30**
Total No. of Azimuth Processors:			**19**

This slide describes each of the processing steps for both range and azimuth processing and summarizes the processor throughput and corresponding processing requirement for each step for the case of the example SAR radar parameters given in the preceding slide.

The effective number of processors required for each step is then determined by dividing that step's processing requirement by the corresponding value of processor throughput. The required total number of processors is then determined by adding up the effective number of processors required at each step over all steps. A processing margin of 30% is then added to this total. The resulting processor totals are derived separately for range and azimuth processing.

Memory Requirement Determination

- **Intermediate Results stored in (16-by-16) bit complex fixed-point to save memory (≈ 90 dB dynamic range)**
- **Range Processors**
  - Total: $16\ P_r\ N_r = 16(11)(20 \times 10^3)$ **= 3.5 MB**
  - Per Processor: $16\ N_r = 16(20 \times 10^3)$ **= 0.32 MB**
- **Azimuth Processors**
  - Total: $20\ N_r\ TW_{az} = 20(20 \times 10^3)(2048)$ **= 819.2 MB**
  - Per Processor: $(20\ N_r\ TW_{az})\ /P_{az} = 819.2\ /19$ **= 43.1 MB**

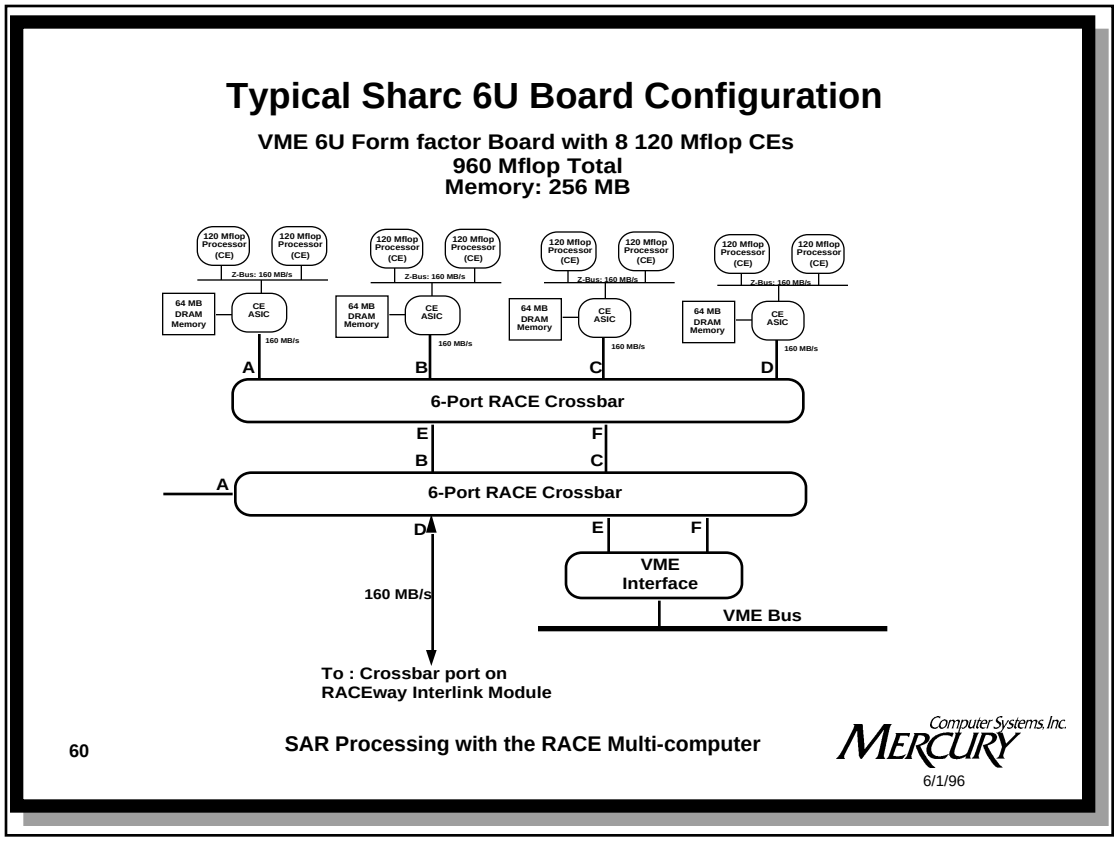
This slide summarizes the memory requirements for both the range and the azimuth processors. The requirements are given both as totals and on a per-processor basis.

Again, it is seen that the requirements of the range processors are nearly negligible in comparison to those of the azimuth processors.

I/O Bandwidth Evaluation

- **Total Data Throughput Q, in samples/s, is the same at all points in the processing pipeline and is equal to the Input Sample Rate**
- **I/O Rate at each processor is (2 x 8) Q / P_s = 16 Q / P_s.**
  - Divide by P_s processors per stage
  - Multiply by 2 for input and output at each processor
  - Multiply by 8 to convert from complex (float) samples to bytes
- **Total data Throughput Rate (8 Q) = 32 MB/sec**
  - is well below RACE single path bandwidth of 160 MB/sec.
- **Total I/O Rate at each processor is negligible compared to single processor path bandwidth of 160 MB/s.**
  - Range Processor: **16 Q / P_r = 16 (4 x 10^6) /11 = 6 MB/s**
  - Azimuth Processor: **16 Q / P_{az} = 16 (4 x 10^6) /19 = 3.4 MB/s**
- **Bottom Line:**
  - **Airborne SAR Data Rates are well below even the RACE single path capability of 160 MB/sec. NO PROBLEM!**

This slide summarizes the I/O bandwidth requirements of the example SAR application. Both the overall data throughput and the total I/O per processor, for both range and azimuth processors, are well below the single-path RACE bandwidth.



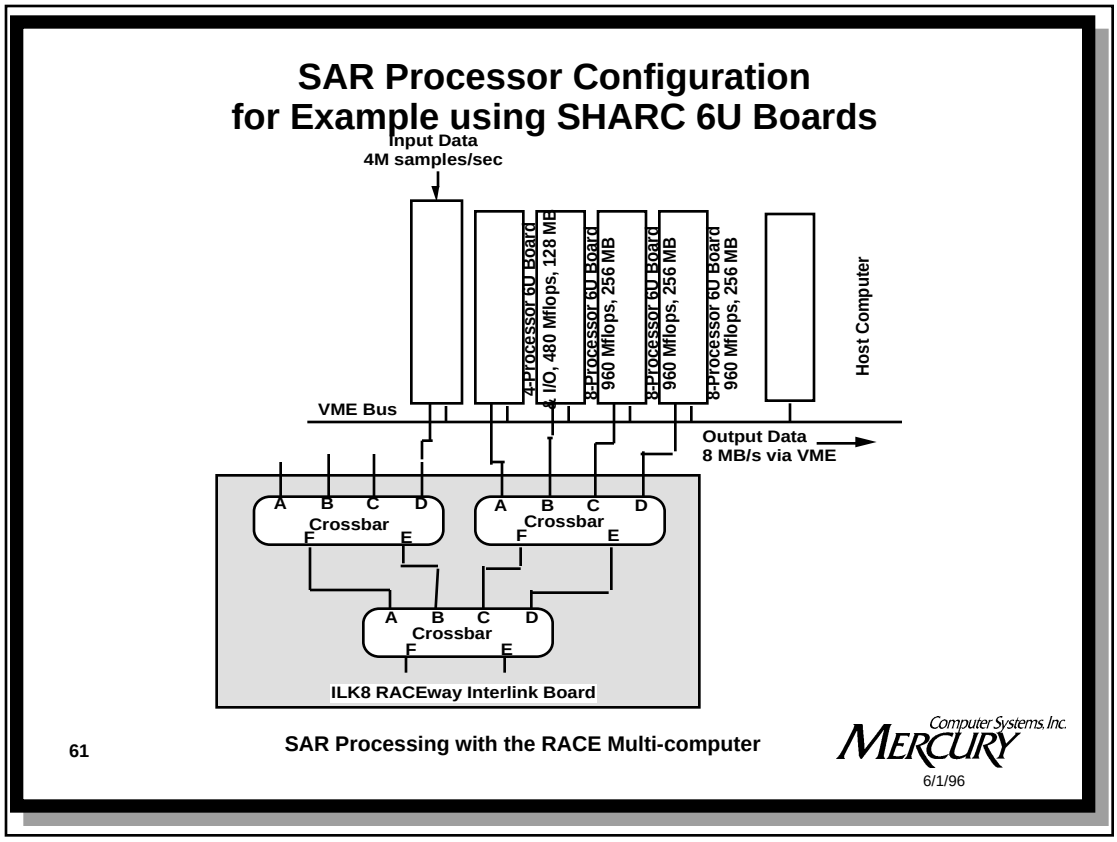
This slide illustrates the organization of both processors (i.e. CEs) and memory on a standard product, 8-processor, 6U form-factor VME board.

The organization consists of four processor nodes, each consisting of two 120 Mflop processors, a shared DRAM memory of 64 MB, and a CE ASIC for memory access arbitration and DMA control. Each of these four nodes is connected to a RACE crossbar port. The single-port bandwidth to/from each node is 160 MB/s.

There are two external data paths off this board; one is a 160 MB/s path to a RACEway Interlink crossbar network attached to the VME chassis backplane, the other path is to the VME bus and can sustain data transfers on the order of 50 MB/s. The latter is prmarily used for communications with the host computer.

In order to optimize the use of the available memory, one of the two processors on each node will be allocated to range processing while the other is allocated to azimuth processing. This means that nearly the entire local node memory can be allocated for use by the azimuth processor.

The first board in the system will have a configuration slightly different from that shown above in that two of the four nodes will be replaced by a high-speed (120 MB/s) input data interface. Hence that board will consist of 4 processors provining 480 Mflops of peak processing power, 128 MB of DRAM memory, and an input interface.



This slide illustrates how the processing requirement for the example application is met using five interconnected boards of the type described in the preceding slide. Four of the five boards contain 8 processors and 256 MB of memory each. The fifth board contains 4 processors, an additional 128 MB, and an input interface. The system has been sized to provide about a 30% growth margin in both processing power and memory.

The five boards are interconnected through an 8-port RACEway Interlink card, ILK8, that plugs into the P2 connector on the VME backplane. As mentioned earlier the interconnection bandwidth provided by this arrangement is 160 MB/s per path which is more than adequate for the application at hand.

Raw data enters the input interface at a rate of 4 Msamples/sec (16 MB/sec) while the processed image output data leaves the system at a rate of 8 MB/s over the VME interface.

Hardware Features of Mercury's RACE Architecture relevant to SAR Processing

- **Efficient Interprocessor Communications**
  - Multiple Concurrent Data Paths between CEs at up to 160 MB/s each
- **Chained DMACapability**
  - Each CE has its own DMA Engine
    - * runs independently of the CPU
  - Allows a CE to transparently distribute a block of data to a set of other CEs without CPU intervention
  - Faciltates "Corner Turning"
- **Heterogeneous Processors**
  - CEs in a given system do not all have to be the same.
  - CE configurations can be optimized for each application.
    - » CEs used for range processing may have much smaller memories than those used for azimuth processing.
  - CEs can be easily upgraded as new types become available

Software Features of Mercury's RACE Architecture relevant to SAR Processing

- **MC/OS Real-Time Operating System**
  - **Compact: only 400 KB per node (including config. tables)**
  - **Modular:**
    - » MCexec - processor's real-time OS
    - » ICS - Interprocessor Communications System
    - » processor-specific nano-kernel
  - **Low Overhead**
- **PAS (Parallel Applications System)**
  - **Applications Interface Layer on Top of ICS**
    - » Automatically distributes SPMD range compression and azimuth compression processes over a designated number of multiple processors.
    - » Automatically performs chained DMA transfers and local transposes required for corner-turning operation.