High Performance Flexible DSP Infrastructure Based on MPI and VSIPL

7th Annual Workshop on High Performance Embedded Computing

MIT Lincoln Laboratory
23-25 Sept 2003

Tom McClean
Lead Member
Engineering Staff

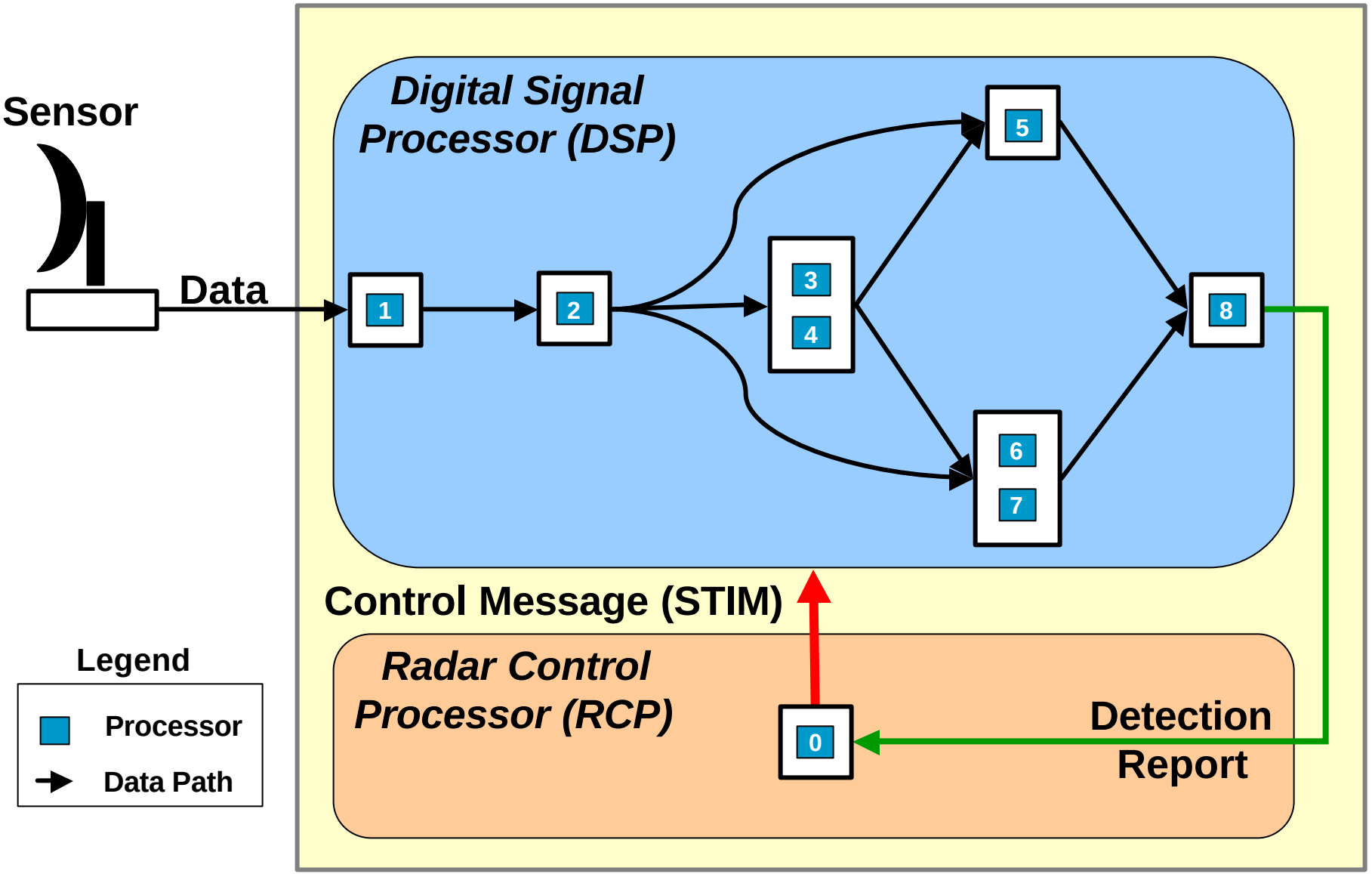
LOCKHEED MARTIN

Hard Real Time DSP Challenge

- ***Develop a Portable and Easily Scalable DSP***
  - ***Portability requires the use of Open Architecture Standards***
    - ***Low Overhead Communication***
      - Message Passing Interface (MPI)
    - ***Vector Processing***
      - Vector Signal Image Processing Library (VSIPL)
    - ***Programming Language (C++)***
  - ***Scalability requires:***
    - ***An Infrastructure which is highly configurable.***
      - Number of Processors
      - Round Robin, Pipeline or Hybrid Data Flow
      - Data Redistribution Support
        - Frees the algorithm designer from the details of the data distribution

Open Architecture Standards allow for Platform Flexibility

Digital Processor Block Diagram



Real Time DSP Solution

- ***DSP Infrastructure Description***
  - ***Flow Graph***
    - ***Defines the Data Flow and Algorithm Mapping to a Network of Processors***
      - Based on a Static Map of DSP processors
      - Infrastructure Supports Multiple Flow Graphs
      - Text File Based (Easily Modified)
      - Loaded during Software Initialization
      - Easy to add algorithms or modify data flow
  - ***MPI Intercommunicators are formed based on Flow Graph information.***
    - ***Provides Stimulus and Data Flow Paths.***
    - ***Redistribution API uses the formed Data Paths.***
  - ***Infrastructure has been tested on Server and Embedded architectures using more than 64 processors.***
    - ***No code modification is needed.***
    - ***DSP recompiled for the particular architecture.***

Infrastructure is Platform Independent

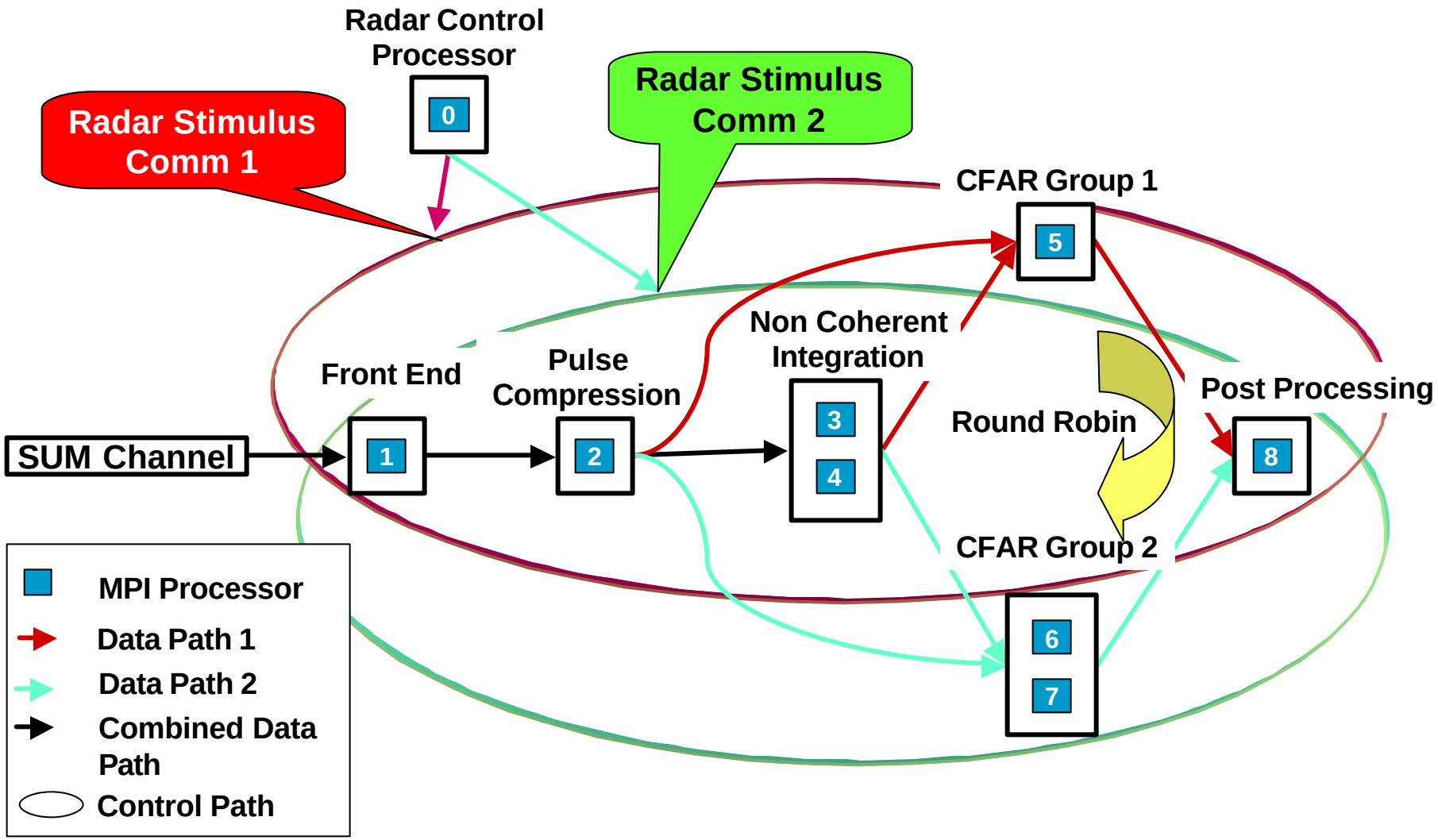
Flow Graph Details

- ***MPI Stimulus and Data flow Communication paths are formed based on information read from text Flow_Graph files during initialization.***

Example DSP Flow Graph							
Processor	*MODE*	*Purpose*	*Group*	*Group Size*	*Group Count*	*Input*	*Output*
0	RCP	STIM	1	1	1	NONE	NONE
1	DSP	FE_SUM	1	1	1	NONE	PC_SUM
2	DSP	PC_SUM	1	1	1	FE_SUM	NCI,CFAR
3, 4	DSP	NCI	1	2	1	PC_SUM	CFAR
5	DSP	CFAR	1	1	2	PC_SUM,NCI	POST_PRO
6, 7	DSP	CFAR	2	2	2	PC_SUM,NCI	POST_PRO
8	DSP	POST_PRO	1	1	1	CFAR	NONE

Reconfiguration does not require code modification

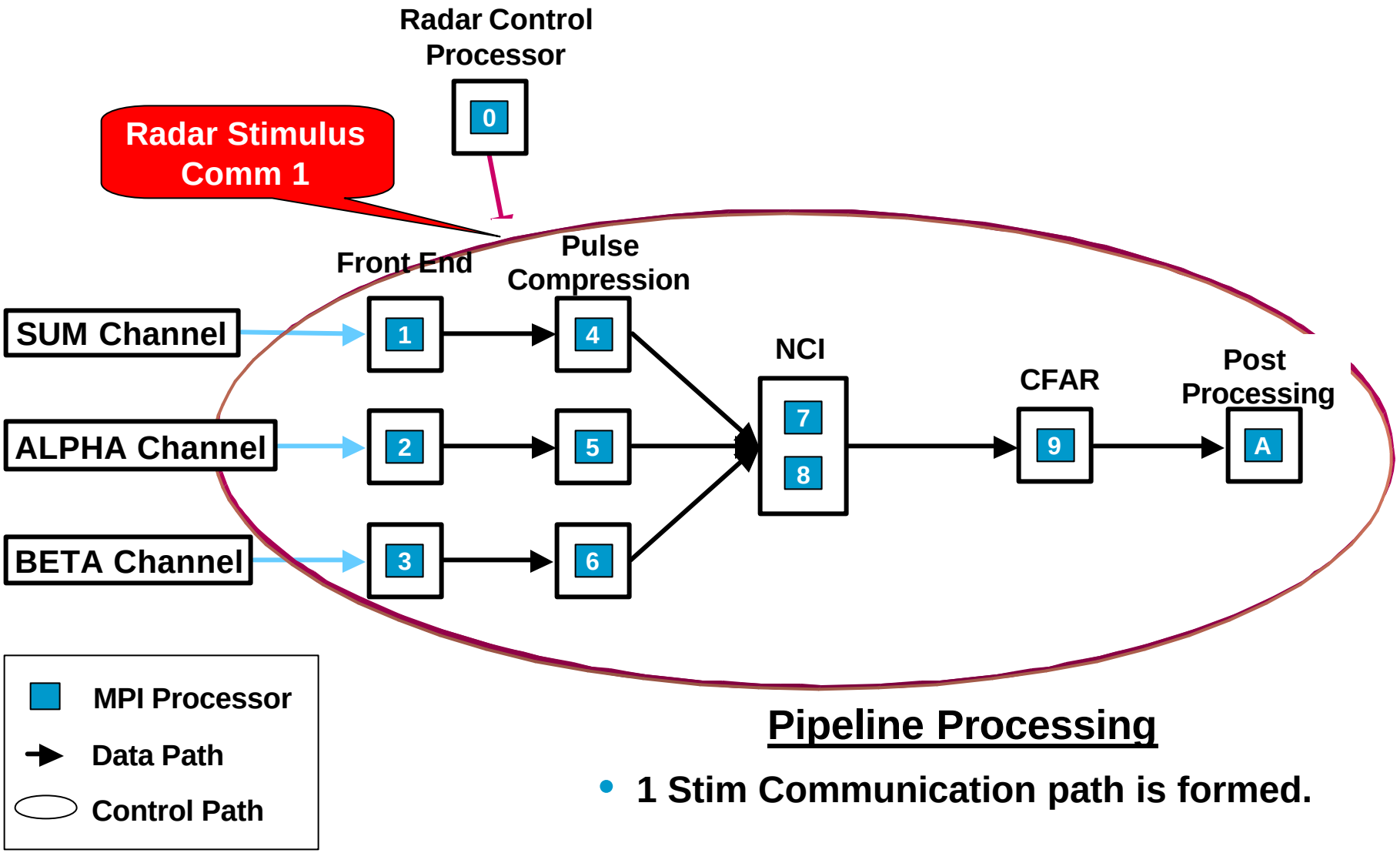
Flow Graph Communicators
Resulting Stim and Data Communication Paths



Pipeline Flow Graph

Pipeline DSP Flow Graph							
Processor	*MODE*	*Purpose*	*Group*	*Group Size*	*Group Count*	*Input*	*Output*
0	RCP	STIM	1	1	1	NONE	NONE
1	DSP	FE_SUM	1	1	1	NONE	PC_SUM
2	DSP	FE_AZ	1	1	1	NONE	PC_AZ
3	DSP	FE_EL	1	1	1	NONE	PC_EL
4	DSP	PC_SUM	1	1	1	FE_SUM	NCI
5	DSP	PC_AZ	1	1	1	FE_AZ	NCI
6	DSP	PC_EL	1	1	1	FE_EL	NCI
7,8	DSP	NCI	1	2	1	FE_SUM, FE_AZ, FE_EL	CFAR
9	DSP	CFAR	1	1	1	NCI	POST_PRO
A	DSP	POST_PRO	1	1	1	CFAR	NONE

Pipeline Processing
Resulting Control and Data Communication Paths



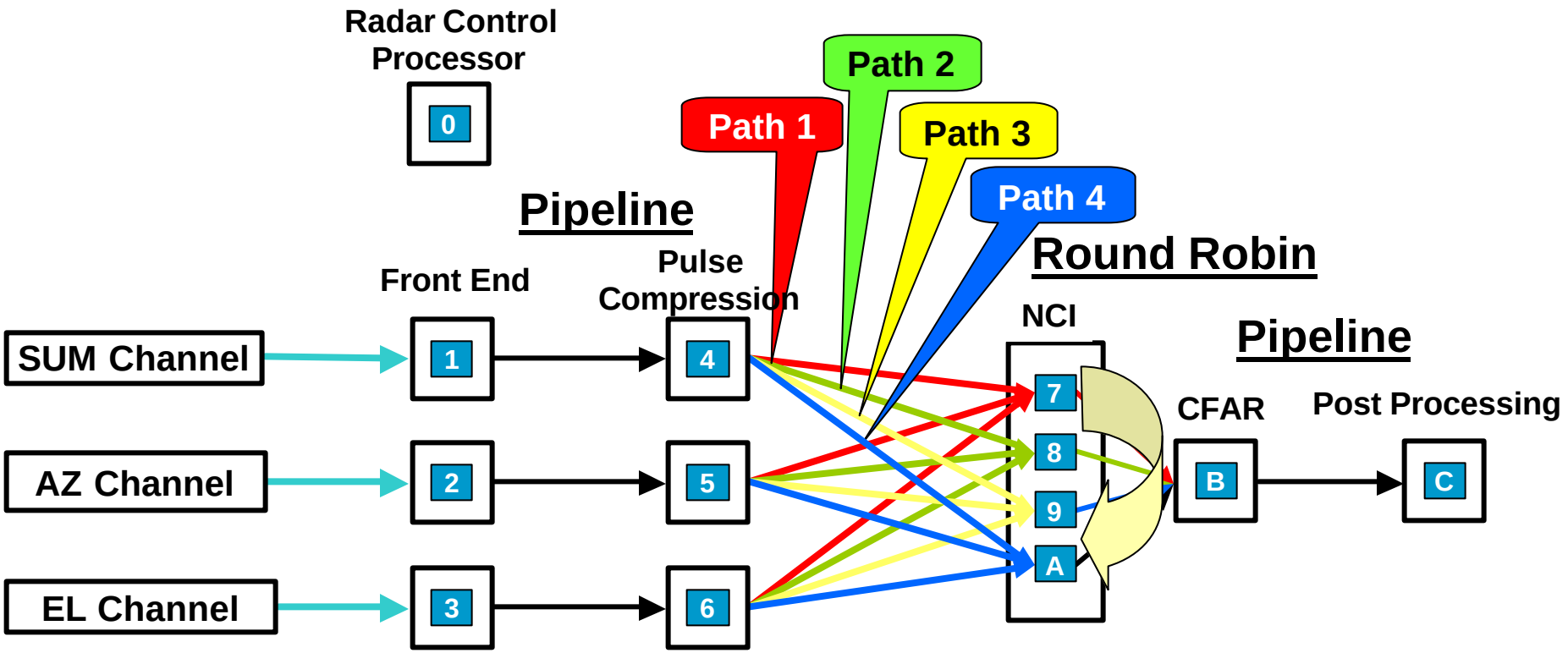
Pipeline Processing

- 1 Stim Communication path is formed.

Hybrid Flow Graph

Hybrid DSP Flow Graph							
Processor	*MODE*	*Purpose*	*Group*	*Group Size*	*Group Count*	*Input*	*Output*
0	RCP	STIM	1	1	1	NONE	NONE
1	DSP	FE_SUM	1	1	1	NONE	PC_SUM
2	DSP	FE_AZ	1	1	1	NONE	PC_AZ
3	DSP	FE_EL	1	1	1	NONE	PC_EL
4	DSP	PC_SUM	1	1	1	FE_SUM	NCI
5	DSP	PC_AZ	1	1	1	FE_AZ	NCI
6	DSP	PC_EL	1	1	1	FE_EL	NCI
7	DSP	NCI	1	1	4	FE_SUM, FE_AZ,FE_EL	CFAR
8	DSP	NCI	1	1	4	FE_SUM, FE_AZ,FE_EL	CFAR
9	DSP	NCI	1	1	4	FE_SUM, FE_AZ,FE_EL	CFAR
A	DSP	NCI	1	1	4	FE_SUM, FE_AZ,FE_EL	CFAR
B	DSP	CFAR	1	1	1	NCI	POST_PRO
C	DSP	POST_PRO	1	1	1	CFAR	NONE

Hybrid Processing
Resulting Stim and Data Communication Paths



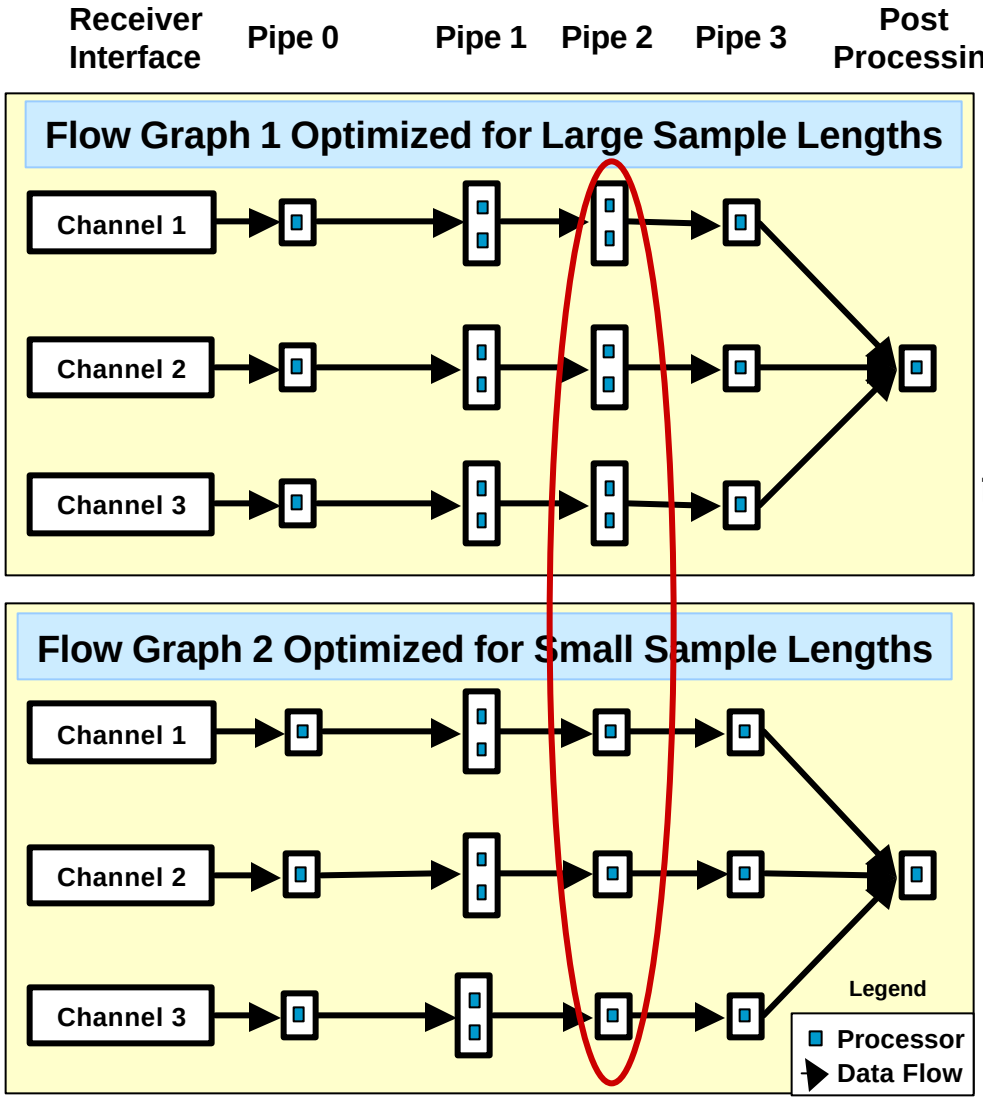
Hybrid Processing

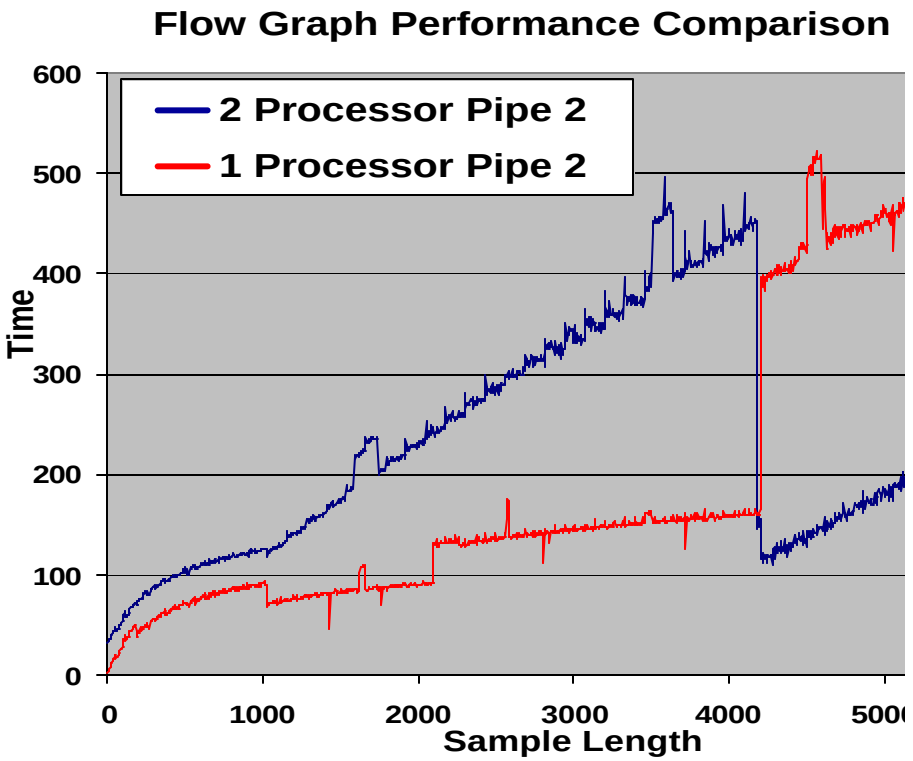
- 4 Distinct Communication paths are formed.
- A path is used per Radar Sequence. (ROUND ROBIN)
- Stim distributor determines which Comm path is in use.
- Stimulus is only distributed to processors that need it.

MPI Processor

Data Path

Multiple Flow Graphs





Each Flow Graph is Optimized for Radar Modes or Data Size

Data Redistribution

- *Data Flow paths are used for Redistribution*
- *Data Redistribution Calls are Layered on MPI*
- *Provide 2D->2D with Overlap and Modulo support*
- *Insulates Algorithm from Redistribution*

Algorithm Pseudo Code Fragment

Data Input From 2 Processors to 3 Processors

VSIPL provides a Platform independent API

Data Output From 3 Processors to 1 Processor

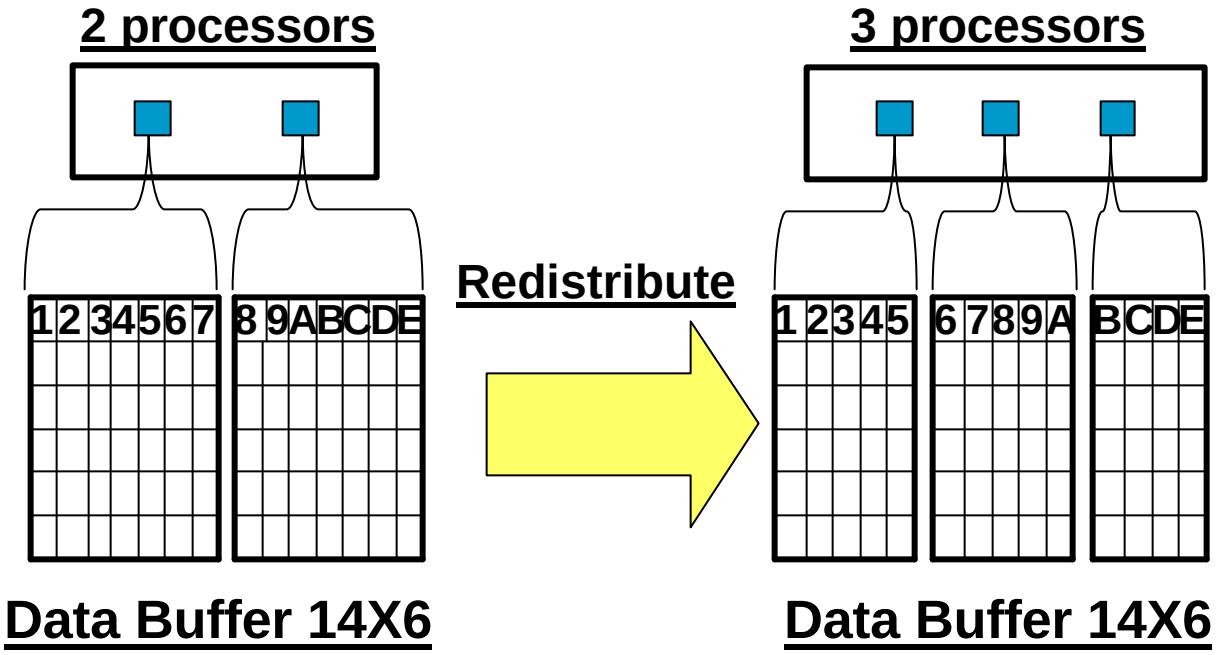
```
// Data input scalable across processors
// Receive Input Data
blocks = Redist( Buffer, 14, 23, 1, 0);

// Perform algorithm on received data
for( int i=0; i<blocks; i++)
{
  vsip_ccfftop_f(…);
  …
}
// Data output scalable across processors
// Send Output Data
blocks = Redist( Buffer, 14, 32, 1, 0);
```

Developer can Concentrate on Algorithm Implementation

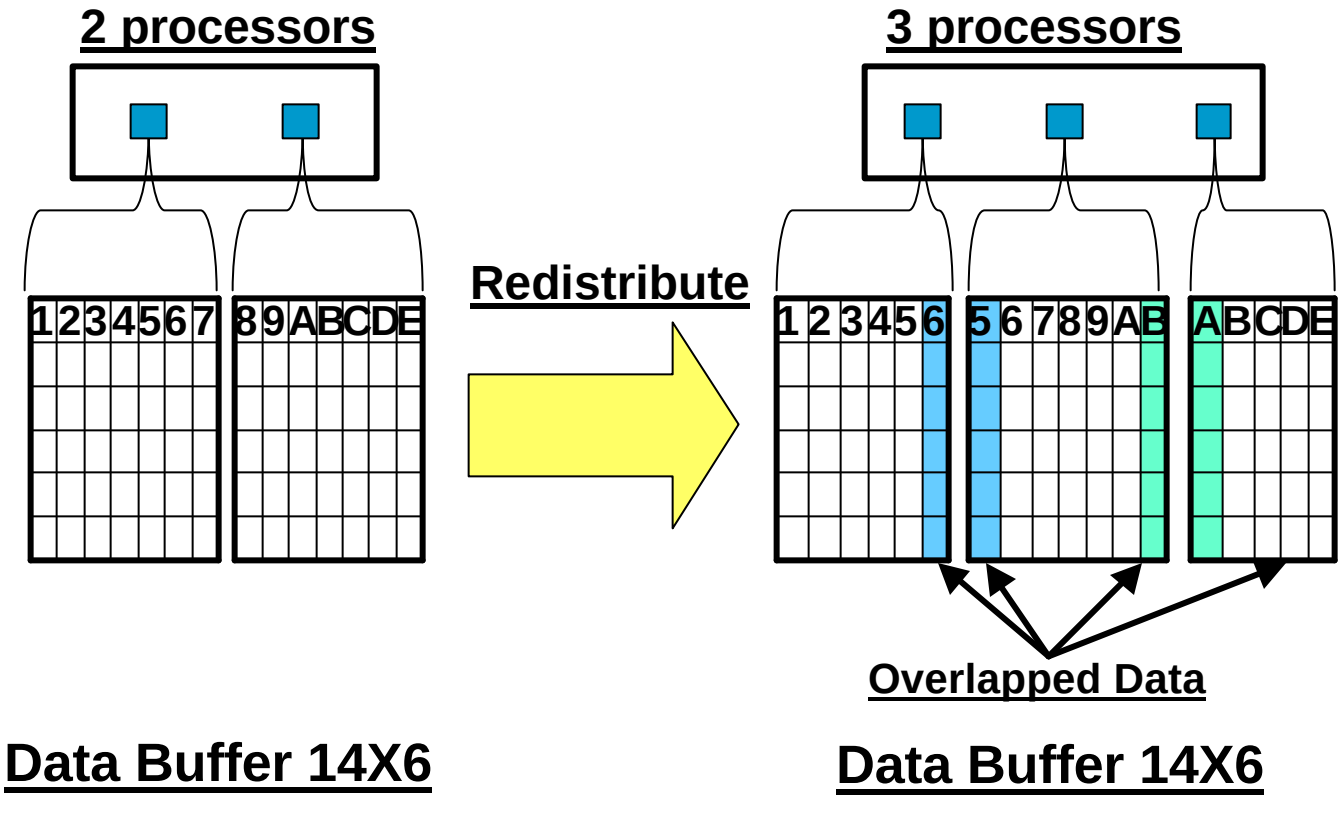
Data Redistribution
Without Overlap

- *Data flow Communication paths are used for Redistribution*
- *Data Redistribution Calls are Layered on MPI*
- *Provide 2D->2D with Overlap and Modulo support*
- *Insulates Algorithm from Redistribution*
  - *Redist(Data_Buffer, Splitable Dimension, Unsplitable Dimension, Modulo,Overlap);*
  - *Redist(Buffer, 14, 6, 1, 0); -Application Redistribution Call*



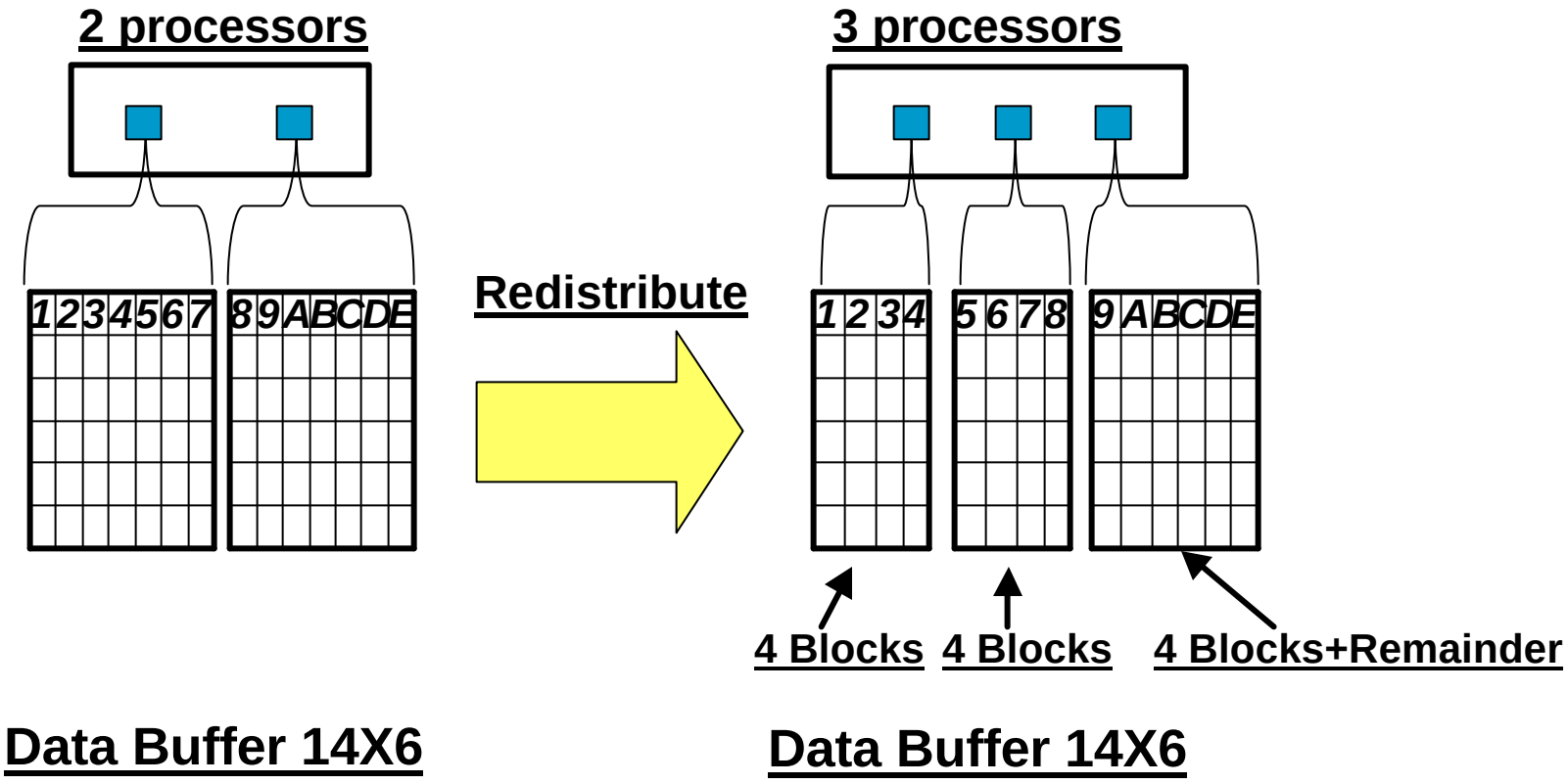
Data Redistribution
With Overlap

- ***Redist(Data_Buffer, Splitable Dimension, Unsplitable Dimension, Modulo, Overlap);***
- ***The Same Call is Made by all 5 Processors***
- ***Redist(Buffer, 14, 6, 1, 1); -Application Redistribution Call With Overlap 1***



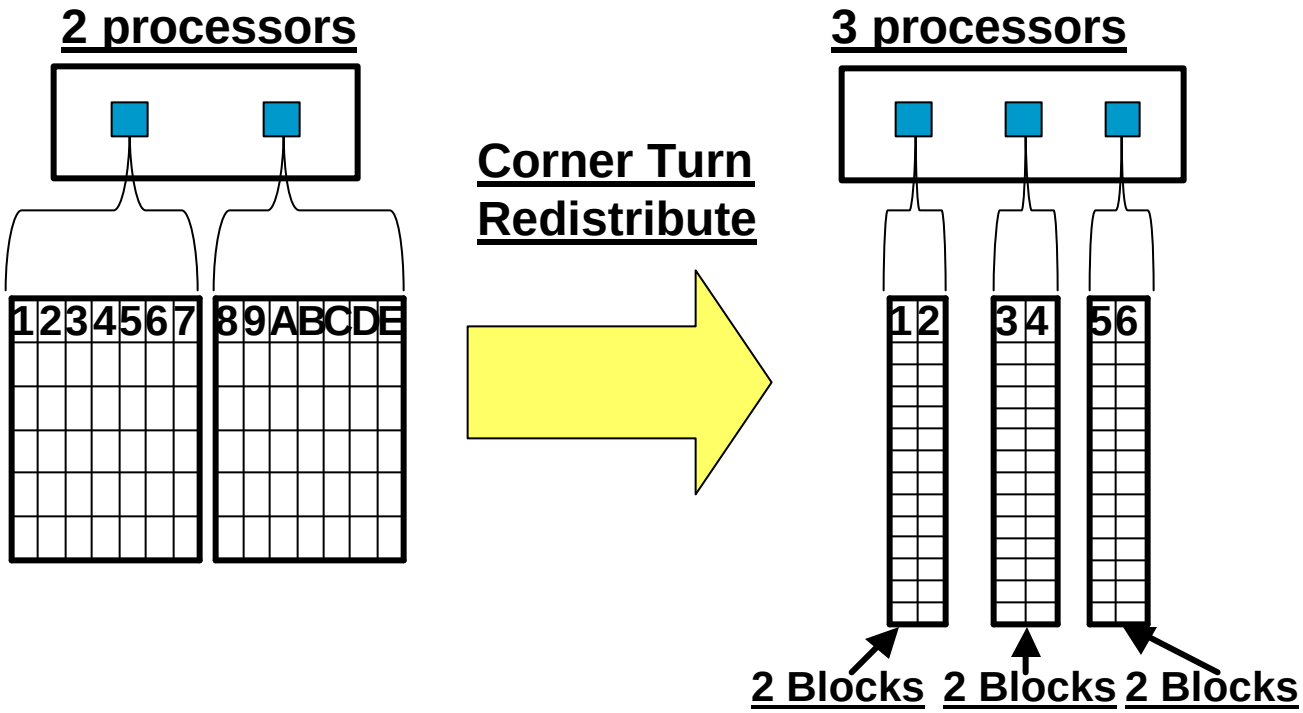
Data Redistribution
With Modulo

- ***Redist(Data_Buffer, Splitable Dimension, Unsplitable Dimension, Modulo, Overlap);***
- ***The Same Call is Made by all 5 Processors***
- ***Redist(Buffer, 14, 6, 4, 0); -Application Redistribution Call With Modulo 4***



Matrix Transpose

- ***Ct_Transfer(Data_Buffer, Splitable Dimension, Unsplitable Dimension, Modulo);***
- ***The Same Call is Made by all 5 Processors***
- ***Ct_Transfer(Buffer, 14, 6, 1); - Application Matrix Transpose***



Summary

- ***DSP Infrastructure:***
  - ***Supports Real-Time High-Performance Embedded Radar Applications***
    - ***Low Overhead***
    - ***Scalable to requirements***
  - ***Built on Open Architecture Standards***
    - ***MPI and VSIPL***
      - Reduces development costs
        - Scalable to applications with minimal changes to software
      - Provides for Platform Independence
  - ***Provides DSP Lifecycle Support***
    - ***Scale DSP from Development to Delivery Without Code Modifications***
    - ***Add Algorithms with Minimal Software Changes***
    - ***Reusable Infrastructure and Algorithms***
    - ***Easily Scale DSP for Various Deployments***

Infrastructure Reduces Development Cost