Simulation of the Communication Time for a Space-Time Adaptive Processing Algorithm on a Parallel Embedded System

by
Jack M. West and John K. Antonio
Texas Tech University
{west, antonio}@ttu.edu

EHPC ‘98:
Workshop on Embedded HPC Systems and Applications
April 3, 1998

Supported by
Rome Laboratory Grant No. F30602-96-1-0098 and DARPA Contract No. F30602-97-2-0297
Outline

• STAP Basics

• Parallelization of STAP

• Impact of Mapping and Scheduling Choices on Communication Delay

• RACE Network Simulator

• Summary
Space-Time Adaptive Processing

- Space-Time Adaptive Processing (STAP) refers to a class of signal processing methods that operate on data collected from a set of sensors over a given time interval.
- STAP simultaneously combines the signals received from an antenna array (spatial domain) and multiple pulse repetition periods (time domain).
- STAP provides improved detection of smaller targets in the presence of ground clutter (overland and littoral environments) and hostile interference (electronic counter measures and jamming).
Typical STAP Processing Flow

Input Data

Pulse Compress

Data Cube

Range

Channels

Pulses

Rotate

Data Cube

Range

Pulses

Channels

Doppler Filter

Data Cube

Range

Pulses

Channels

Beam Outputs

Data Cube

Range

Pulses

QR Decomposition

Beamform

Weights

Steering Vectors
Outline

• STAP Basics

• Parallelization of STAP

• Impact of Mapping and Scheduling Choices on Communication Delay

• RACE Network Simulator

• Summary
1. Partition STAP data cube over a 2-D process set.
2. Process the contiguous dimension.
3. Re-partition the data cube before processing the next dimension.
4. Rotate the newly distributed data to make the next dimension sequential in memory.
5. Repeat steps 1 through 4 before each processing phase.
Illustration of STAP Data Cube Partitioning

Pulse Compression Partitioning with range dimension whole.

Doppler Filtering Partitioning with pulses dimension whole.
STAP Data Cube Re-Partitioning

- Re-Partitioning involves exchanging data with the next whole dimension.

- Interprocessor Communication is required between processors in the same row.
STAP Data Cube Re-Partitioning

QR-D, Back-Substitution, Beamforming

Re-Partitioning
Apply 3x4 Process Set:

QR-Decomposition & Back-Substitution

Beamforming
Outline

• STAP Basics

• Parallelization of STAP

• Impact of Mapping and Scheduling Choices on Communication Delay

• RACE Network Simulator

• Summary
SHARC Compute Node Architecture

RACEway Interface

SHARC Processor

RACEway Mapping Logic

OS Support Hardware

3-Way Data Switch

ECC Logic

Performance Metering

DMA Controller

CN ASIC

DRAM

RACEway Interface
Data Set Re-Partitioning
with raster ordering in the channel dimension

Required Data Transfers
Total Message Size Count
= 36 units
Data Set Re-Partitioning
with raster ordering in the pulse dimension

Required Data Transfers
Total Message Size Count = 20 units
STAP Data Cube Re-Partitioning

Data Re-Distribution Mapping

Network Interconnection Configuration

6-Port Crossbar

Pulse Compression

Pulses Range

Channel

Pulses Range

Doppler Filtering

Required Data Transfers

IPC
Scheduling Communications

Outgoing Message Queues

6-Port Crossbar

- **A** (3)
- **B** (3)
- **C** (4)
- **D** (3)
- **E** (4)
- **F** (3)

**Minimum Communication Time:**

\[ T_{\text{min}}(A, B, C, D) = T_c(A) = T_c(B) \]

\[ T_c(A) = T_{\text{outgoing}} + T_{\text{incoming}} = (3 + 4) + (3 + 4) = 14 \text{ network cycles} \]

**Actual Communication Time**

\[ T(0) = \max[T_p(A), T_p(D)] = 3 \text{ network cycles (n.c.)} \]

\[ T(3) = T_p(B) = 3 \text{ n.c.} \]

\[ T(6) = T_p(E) = 4 \text{ n.c.} \]

\[ T(10) = T_p(C) = 4 \text{ n.c.} \]

\[ T(14) = T_p(F) = 3 \text{ n.c.} \]

\[ T_c = 17 \text{ network cycles} \]
Scheduling Communications

Exchange Messages C and F in CN Queue

6-Port Crossbar

Actual Communication Time

\[ T(0) = \max \{ T_p(\text{A}), T_p(\text{F}) \} \]
\[ = 3 \text{ network cycles (n.c.)} \]

\[ T(3) = \max \{ T_p(\text{B}), T_p(\text{D}) \} \]
\[ = 3 \text{ n.c.} \]

\[ T(6) = T_p(\text{E}) = 4 \text{ n.c.} \]

\[ T(10) = T_p(\text{C}) = 4 \text{ n.c.} \]

\[ T_c = 14 \text{ network cycles} \]

Outgoing Message Queues
Predicting communication time at each phase depends on many factors:

- Number of CNs used.
- Size of the data cube.
- Topology of the interconnection network.
- Data decomposition scheme used (mapping).
- Scheduling of the communications.
Outline

- STAP Basics
- Parallelization of STAP
- Impact of Mapping and Scheduling Choices on Communication Delay
- RACE Network Simulator
- Summary
Some Features of the RACE Network

1. 40Mhz clock, 32 bit data paths, 2048 byte circuit-switched packets.

2. Contention resolved using priorities
   - User-programmable message priority
   - Hardware priority assigned at each crossbar along a path (based on complex connection rules)

3. A message with higher priority preempts (suspends) a lower priority message (active or inactive) to gain control of a crossbar port

SIMULATOR DATA FLOW

User Interface
- Network Size
- STAP Data Cube Parameters
- Process Set Configuration
- Chaining Options
- Message Ordering

Network Object

- Build Network
- Build Network Routing Table
- Build - Order Message Traffic
- Build STAP Data Cube
- Build Process Set
- Simulate Message Traffic

User Interface
- Display the Timing Results
Network Object

Network Object Methods

- Dynamic Network Construction
- Dynamic Routing Table Creation
- Dynamic CN Message Traffic Generation
- Simulates “Corner-Turn” Traffic

Random Scan Object

- Generates Pseudo-Random CN Scan Ordering

Simulation Clock Object

- Based on Network Clock Frequency
- Data Transfer Rate Equates to Effective Network Bandwidth
CROSSBAR OBJECT

Crossbar Object
- Two Parent Port Connections
- Four Child Port Connections
- Internal Switch Connections
- Terminal Crossbars Contain CN Connections

Crossbar Object Methods
- Implements Hardware Priority Arbitration
  - Top-Level Algorithm
  - Standard Algorithm
- Query Port Status
- Routes Packets to Next Location
- Allocates and Frees Port Connections and Connected Link Objects
- Transmits Packet Data

Link Object
- Connects Crossbars or Compute Nodes
- Link Status: Occupied or Free
**Compute Node Object**

- Processor Information
- Outgoing and Incoming Message Queues
- Packet Stack
  - Packets are self-routing

**Outgoing Message Queue**
- Message 1
- Message 2
- Message 3
- ...

**Packet Stack**
- Packet 1
- Packet 2
- Packet 3
- Packet 4
- ...

**Compute Node Methods**
- Manages Outgoing and Incoming Message Queues
- Manages Packet Stack
- Explodes the Top Outgoing Message into Packets of Size 2048 or Less
- Handles DMA Chaining of Packets
- Establishes Path Through Network and Transmits Packet Data
Outline

• STAP Basics

• Parallelization of STAP

• Impact of Mapping and Scheduling Choices on Communication Delay

• RACE Network Simulator

• Summary
• Designed and implemented network simulator that models the impact of data mapping and scheduling on the performance of a STAP

• Implemented in OO paradigm using JAVA (approximately 6000 LOC)

• Implementation of all network simulation objects completed. Currently testing and completing user interface objects.

• Can be used with any parallel application that has distinct and definable communication phases.

• Very fast execution time - Portable - Available on-line soon!