High Performance Flexible DSP Infrastructure Based on MPI

Tom McClean and Steve Shank
Lockheed Martin, Naval Electronics and Surveillance Systems (NESS)
Phone: (856)-914-6376
Fax: (856)-914-6815
Email: tom.mcclean@lmco.com
Email: steve.f.shank@lmco.com

Abstract:
Lockheed Martin has developed a platform independent, scalable and reconfigurable Digital Processor (DP) infrastructure for use in multiprocessor environments. This infrastructure is in use within the Small System Processor (SSP) program. This infrastructure provides communication, data flow, processor/algorithm scaling and configuration flexibility. All aspects of communication and processing are reconfigurable without the need to recompile. Pipeline, round robin, or hybrid processing architectures are supported, as well as modifying the number of processors without the need to recompile. This flexibility is provided by the use of text “flow graph” files, which describe a static processor mapping. Multiple flow graphs are supported.

A non-blocking multicast API is also provided. This is used to distribute the DP Stimulus messages to only the processors that are required to participate in processing.

The communication infrastructure provides an efficient mechanism, which decouples algorithm development from the specific details of the data distribution. Algorithm data flow routines support redistributing data from M to N processors with or without data overlap or minimum block sizes. Also provided are M to N corner turn and algorithm corner turn routines. Blocking and Non Blocking API’s are provided.

This infrastructure is highly portable. The infrastructure was developed on CSPI 2841 multiprocessors using MPI as the underlying communication API and VSIPL as the Vector math library. Because it is based on industry standard API’s, this infrastructure can be run on any platform that supports these API’s. This has been validated on Server Class as well as Embedded platforms. No change to code was made, just a recompile for the particular platform.
**High Performance Flexible DSP Infrastructure Based on MPI**

**Lockheed Martin, Naval Electronics and Surveillance Systems (NESS)**

See also ADM001694, HPEC-6-Vol 1 ESC-TR-2003-081; High Performance Embedded Computing (HPEC) Workshop (7th), The original document contains color images.
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
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
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.**

<table>
<thead>
<tr>
<th>Processor</th>
<th>MODE</th>
<th>Purpose</th>
<th>Group</th>
<th>Group Size</th>
<th>Group Count</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RCP</td>
<td>STIM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>1</td>
<td>DSP</td>
<td>FE_SUM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>NONE</td>
<td>PC_SUM</td>
</tr>
<tr>
<td>2</td>
<td>DSP</td>
<td>PC_SUM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>FE_SUM</td>
<td>NCI,CFAR</td>
</tr>
<tr>
<td>3, 4</td>
<td>DSP</td>
<td>NCI</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>PC_SUM</td>
<td>CFAR</td>
</tr>
<tr>
<td>5</td>
<td>DSP</td>
<td>CFAR</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>PC_SUM,NCI</td>
<td>POST_PRO</td>
</tr>
<tr>
<td>6, 7</td>
<td>DSP</td>
<td>CFAR</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>PC_SUM,NCI</td>
<td>POST_PRO</td>
</tr>
<tr>
<td>8</td>
<td>DSP</td>
<td>POST_PRO</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>CFAR</td>
<td>NONE</td>
</tr>
</tbody>
</table>

Reconfiguration does not require code modification
Flow Graph Communicators
Resulting Stim and Data Communication Paths
## Pipeline Flow Graph

### Pipeline DSP Flow Graph

<table>
<thead>
<tr>
<th>Processor</th>
<th>MODE</th>
<th>Purpose</th>
<th>Group</th>
<th>Group Size</th>
<th>Group Count</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RCP</td>
<td>STIM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>1</td>
<td>DSP</td>
<td>FE_SUM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>NONE</td>
<td>PC_SUM</td>
</tr>
<tr>
<td>2</td>
<td>DSP</td>
<td>FE_AZ</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>NONE</td>
<td>PC_AZ</td>
</tr>
<tr>
<td>3</td>
<td>DSP</td>
<td>FE_EL</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>NONE</td>
<td>PC_EL</td>
</tr>
<tr>
<td>4</td>
<td>DSP</td>
<td>PC_SUM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>FE_SUM</td>
<td>NCI</td>
</tr>
<tr>
<td>5</td>
<td>DSP</td>
<td>PC_AZ</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>FE_AZ</td>
<td>NCI</td>
</tr>
<tr>
<td>6</td>
<td>DSP</td>
<td>PC_EL</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>FE_EL</td>
<td>NCI</td>
</tr>
<tr>
<td>7,8</td>
<td>DSP</td>
<td>NCI</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>FE_SUM, FE_AZ, FE_EL</td>
<td>CFAR</td>
</tr>
<tr>
<td>9</td>
<td>DSP</td>
<td>CFAR</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>NCI</td>
<td>POST_PRO</td>
</tr>
<tr>
<td>A</td>
<td>DSP</td>
<td>POST_PRO</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>CFAR</td>
<td>NONE</td>
</tr>
</tbody>
</table>
Pipeline Processing
Resulting Control and Data Communication Paths

- Radar Stimulus Comm 1
- Front End
- Pulse Compression
- NCI
- CFAR
- Post Processing

SUM Channel
ALPHA Channel
BETA Channel

MPI Processor
Data Path
Control Path

Pipeline Processing
• 1 Stim Communication path is formed.
### Hybrid DSP Flow Graph

<table>
<thead>
<tr>
<th>Processor</th>
<th>MODE</th>
<th>Purpose</th>
<th>Group</th>
<th>Group Size</th>
<th>Group Count</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RCP</td>
<td>STIM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>1</td>
<td>DSP</td>
<td>FE_SUM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>NONE</td>
<td>PC_SUM</td>
</tr>
<tr>
<td>2</td>
<td>DSP</td>
<td>FE_AZ</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>NONE</td>
<td>PC_AZ</td>
</tr>
<tr>
<td>3</td>
<td>DSP</td>
<td>FE_EL</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>NONE</td>
<td>PC_EL</td>
</tr>
<tr>
<td>4</td>
<td>DSP</td>
<td>PC_SUM</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>FE_SUM</td>
<td>NCI</td>
</tr>
<tr>
<td>5</td>
<td>DSP</td>
<td>PC_AZ</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>FE_AZ</td>
<td>NCI</td>
</tr>
<tr>
<td>6</td>
<td>DSP</td>
<td>PC_EL</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>FE_EL</td>
<td>NCI</td>
</tr>
<tr>
<td>7</td>
<td>DSP</td>
<td>NCI</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>FE_SUM, FE_AZ, FE_EL</td>
<td>CFAR</td>
</tr>
<tr>
<td>8</td>
<td>DSP</td>
<td>NCI</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>FE_SUM, FE_AZ, FE_EL</td>
<td>CFAR</td>
</tr>
<tr>
<td>9</td>
<td>DSP</td>
<td>NCI</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>FE_SUM, FE_AZ, FE_EL</td>
<td>CFAR</td>
</tr>
<tr>
<td>A</td>
<td>DSP</td>
<td>NCI</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>FE_SUM, FE_AZ, FE_EL</td>
<td>CFAR</td>
</tr>
<tr>
<td>B</td>
<td>DSP</td>
<td>CFAR</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>NCI</td>
<td>POST_PRO</td>
</tr>
<tr>
<td>C</td>
<td>DSP</td>
<td>POST_PRO</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>CFAR</td>
<td>NONE</td>
</tr>
</tbody>
</table>
Hybrid Processing

Resulting Stim and Data Communication Paths

- 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.
Multiple Flow Graphs

Flow Graph 1 Optimized for Large Sample Lengths
- Channel 1
- Channel 2
- Channel 3

Flow Graph 2 Optimized for Small Sample Lengths
- Channel 1
- Channel 2
- Channel 3

Legend
- Blue: Processor
- Red: Data Flow

Flow Graph Performance Comparison
- Blue: 2 Processor Pipe 2
- Red: 1 Processor Pipe 2

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**

```
// 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);
```

VSIPL provides a Platform independent API

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 Buffer 14X6

![Diagram showing redistribution from 2 processors to 3 processors.](image)
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

- \textbf{Redist}( \text{Data\_Buffer, Splitable Dimension, Unsplitable Dimension, Modulo, Overlap});
- \textbf{The Same Call is Made by all 5 Processors}
- \textbf{Redist}( \text{Buffer, 14, 6, 4, 0); -Application Redistribution Call With Modulo 4}
Matrix Transpose

- \texttt{Ct\_Transfer(Data\_Buffer, Splitable\ Dimension, Unsplitable\ Dimension, Modulo)};
- \textit{The Same Call is Made by all 5 Processors}
- \texttt{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