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
Digital Processor Block Diagram

Sensor

Digital Signal Processor (DSP)

Control Message (STIM)

Radar Control Processor (RCP)

Legend

- Processor
- Data Path

Detection Report

Legend:
- Processor
- Data Path
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

<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

Legend
- MPI Processor
- Data Path 1
- Data Path 2
- Combined Data Path
- Control Path

Radar Stimulus Comm 1
SUM Channel

Radar Control Processor

Radar Stimulus Comm 2

Front End
Pulse Compression

Non Coherent Integration

Round Robin

Post Processing

CFAR Group 1

CFAR Group 2

Data Path 1
Combined Data Path
Data Path 2

1 2 3 4 5 6 7 8
### 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,</td>
<td>CFAR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FE_AZ,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FE_EL</td>
<td></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
- SUM Channel
- ALPHA Channel
- BETA Channel
- Radar Control Processor
- Front End
- Pulse Compression
- NCI
- CFAR
- Post Processing
- 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

Flow Graph Performance Comparison

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 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**

**Data Input From 2 Processors to 3 Processors**

**Data Output From 3 Processors to 1 Processor**

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

- \textit{Redist}(\textit{Data\_Buffer, Splitable Dimension, Unsplitable Dimension, Modulo, Overlap});
- \textit{The Same Call is Made by all 5 Processors}
- \textit{Redist}(\textit{Buffer, 14, 6, 1, 1}); -Application Redistribution Call With Overlap 1

\begin{itemize}
  \item \textbf{2 processors}
  \begin{itemize}
    \item \textbf{Data Buffer 14X6}
  \end{itemize}

  \item \textbf{3 processors}
  \begin{itemize}
    \item \textbf{Data Buffer 14X6}
    \item Overlapped Data
  \end{itemize}
\end{itemize}
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

Data Buffer 14X6

Redistribute

Data Buffer 14X6
Matrix Transpose

- \texttt{Ct\_Transfer(Data\_Buffer,\ Splitable\ Dimension,\ Unsplitable\ Dimension,\ Modulo)};
- \textbf{The Same Call is Made by all 5 Processors}
- \texttt{Ct\_Transfer(Buffer, 14, 6, 1)}; - Application Matrix Transpose

\begin{itemize}
\item \textbf{Corner Turn Redistribute}
\item Data Buffer 14X6
\item Data Buffer 6X14
\end{itemize}
Summary

• DSP Infrastructure:
  – Supports Real-Time High-Performance Embedded Radar Applications
    • Low Overhead
    • Scalable to requirements
  – Built on Open Architecture Standards
    • MPI and VSIP
      – 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