Optimizing the performance and portability of multicore DSP platforms with a scalable programming model supporting the Multicore Association’s MC-API

June 26, 2012

Debbie Greenstreet
• Strategic Marketing Director Multicore Processors Business, Texas Instruments
• Driving Texas Instruments’ leading edge multicore processors and base station SoCs into the market
• VoIP industry pioneer and evangelist
dgreenstreet@ti.com

Sven Brehmer
• CEO and founder PolyCore Software, Inc.
• MCA Board member
• Chairman of MC-API working group
• Significant contributor on MR-API specification
sven.brehmer@polycoresoftware.com

Markus Levy
• President of Multicore Association
• Chairman of Multicore Technical Conference and Expo
• Founder and President of EEMBC
markus.levy@multicore-association.org
Presentation Overview

- Applications & Market Requirements
- Solution – Multicore SoC Platform
- Programming Multicore – Challenges
- Application Perspective
- Message Passing Programming Model
- MCA & Standards
- Method & Use Case
- Tools
- Portability Example
- Conclusions
High Performance Multicore Applications

Mission Critical

Test and Automation

High Performance Computing, Imaging

Video and Audio Infrastructure

Video Surveillance

Wireless Base Stations
Multicore Platform Market Requirements

- **High performance computing**
  - Increased computation with lower latency

- **Green solutions**
  - Reduced power consumption

- **Consolidation**
  - Reduced size, cost of solution

- **Scalability**
  - Solution offerings with varying performance and cost points

- **Time to market**
  - Optimized development and test costs and schedules

- **Application portability**
  - Migrate and grow to new platforms
<table>
<thead>
<tr>
<th>Solution - Multicore SoC Platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Higher Performance, Lower Power Consumption</strong></td>
</tr>
<tr>
<td>• Multiple Cores, Multiple Types, CPU, DSP, Accelerator</td>
</tr>
<tr>
<td><strong>Reduced BoM</strong></td>
</tr>
<tr>
<td>• Higher Level of Integration</td>
</tr>
<tr>
<td><strong>Scalability</strong></td>
</tr>
<tr>
<td>• Multicore and Multichip</td>
</tr>
<tr>
<td><strong>Time to Market</strong></td>
</tr>
<tr>
<td>• Software Platform Abstraction, Common Platform, Tools</td>
</tr>
<tr>
<td><strong>Application Portability</strong></td>
</tr>
<tr>
<td>• Software Platform Abstraction</td>
</tr>
</tbody>
</table>
TI’s KeyStone Multicore Architecture and Feature Palette

- Enables the creation of common platforms for efficient solutions
- Includes software and tools for optimal time to market
Chip Infrastructure
  • TeraNet
  • Multicore Navigator
  • Multicore Shared Memory
  • Device Security

CorePacs
  • ARM, DSP/VSP

Multicore Shared Memory
  • Fast Access
  • Cache Coherency

AccelerationPacs
  • Packet Processing
  • Security Processing
  • Radio Processing

High Performance I/O
  • Integral Switching

KeyStone Multicore Architecture

Multicore Navigator
  Queue Manager

TeraNet

ARM CorePacs

C66x CorePacs

Multicore Shared Memory Controller

AccelerationPac

I/O
Multicore Navigator

Software Program

Hardware automatically queues tasks to available resources

Local (same chip)

Remote (another chip in the same cluster)

Accelerator

DSP / VSP Core

ARM Core

Multicore Navigator Hardware Assist Enables

• Scalable Software
• Device Pooling
Programming Multicore - Challenges

- Concurrency
  - Multiple Processors – Data Sharing

- Complexity
  - Cores per chip/board/system, accelerators, operating systems, transports, memory architectures.

- Optimizing Performance - Acceleration
  - Programmable cores – DSPs, data sharing, algorithm specific compute engines.

- Portability
  - Multiple Platforms – Same Application

- Knowledge Base
  - Well Known Programming Model
Software/Hardware Infrastructure Model

- Software Application
- Abstraction
- Software Development Kit
- Multicore Navigator
  - Queue Manager
- C66x CorePacs
- ARM CorePacs
- TeraNet
- Multicore Shared Memory Controller
- AccelerationPac
- I/O

- Software Development Kit
Application Perspective

- Same Application – Multiple Platforms
  - Programming model required

- Standardized Programming Model

- Access to Acceleration

- Tools Abstraction “Hides” Platform Specifics
  - Hardware Components
  - Communications Infrastructure

- Application Portability
Suitable Programming Model Message Passing

- Tried and Tested
- Scales Indefinitely
- Homogeneous, Heterogeneous, Accelerators
- Standardized
- Function Isolation and Natural Synchronization
- By Reference or Movement
- Large Knowledge Base
Multicore Association Working Groups

• Completed
  • Multicore Communications API (MC/API) 2.0 – 2000+ Downloads
  • Multicore Resource Management API (MR/API) – almost 1000 Downloads

• Final Review Stages
  • Multicore Programming Practices Guide (MPP)

• Currently Active
  • Tools Infrastructure Working Group (TIWG)
    • Mechanisms exist to support tool interoperability for embedded applications that utilize a single IDE, but today no standard exists to support the embedded multicore environment.
  • Multicore Virtualization Working Group (MVWG)
    • Working on profiling different processor virtualization features.
  • MC/API 2.x
    • Working on ‘zero copy’ functionality (including bidirectional interaction between ‘application and application’ using shared memory and bidirectional interaction between ‘application and driver’) and interoperability.
  • Multicore Task Management API (MT/API)
Multicore Communications API

**Flexibility**

MCAPi Node 1
- endpoint <0,1,1>
- attributes
- port 1
- endpoint <0,1,2>
- attributes
- port 2

connectionless message
- to <0,2,1>

MCAPi Node 2
- endpoint <0,2,1>
- port 1
- attributes

**Messages - Connectionless**
Flexible, less configuration
Blocking and non-blocking
Prioritized messages
*Connectionless more flexibility*

**Node:**
Processes, threads, hardware accelerator or some other thread-of-control abstraction

**Efficiency**

MCAPi Node 1
- endpoint <0,1,1>
- attributes
- port 1

MCAPi Node 3
- endpoint <0,3,1>
- port 1
- attributes

packet channel

MCAPi Node 4
- endpoint <0,4,1>
- attributes
- port 1

MCAPi Node 4
- endpoint <0,3,2>
- attributes
- port 2

**Packets & Scalars - Connected**
More efficient, more configuration
Blocking and non-blocking packets
Blocking scalars
*Connected more efficiency*
Multicore Software Abstraction

Software Application

Tools

Software Development Kit

MCAPI

Multicore Navigator
Queue Manager

C66x CorePacs

ARM CorePacs

TeraNet

Multicore Shared Memory Controller

AccelerationPac

I/O

C66x CorePacs

ARM CorePacs
Multicore Enable

1. Communication Encapsulate Functions
   - Pass Parameters

   Single processor
   - function_1(par)
   - function_2(par)
   - function_3(par)
   - function_4(par)
   - function_5(par)

2. Distribute Functions

   Dual-core
   - function_1(par)
   - function_2(par)
   - mcapi_msg_send
   - pass function_3()
   - Parameters
   - mcapi_msg_recv
   - function_3(par)
   - function_4(par)
   - function_5(par)
Program Model, Use Case: Upgrade Path

Encapsulate Programs

Consolidate

Same MCAPI Enabled Application Source Code

Consolidate & Optimize

8 Cores
Poly-Platform – MCAPI Tool

- Instant MCAPI
  - Topology aware
  - No hand coding

```c
mcapi_msg_send(endpoint[LOCAL_DOMAIN][LOCAL_NODE][1-1], endpoint[0][1][1-1], (void*)
if (mcapi_status != MCAPI_SUCCESS) {
    printf("Error on blocking send: %s\n", mcapi_display_status(mcapi_status, status_message sendError++;
}
mcapi_msg_recv(endpoint[LOCAL_DOMAIN][LOCAL_NODE][1-1], (void*) (void*)recv_data_buffer
if (mcapi_status != MCAPI_SUCCESS) {
    printf("Error on blocking recv: %s\n", mcapi_display_status(mcapi_status, status_message receiveError++;
}
```
Tools Integration

- One Stop Solution
Tools – Platform Mapping
Tools - Accelerators

- Shared Memory
  - DMA support

- sRIO
Portability Example - Application

• Same Application – Two Platforms
  – X86, Win32
  – C66xx, DSP/BIOS

```c
while(1) {
    // alternate between blocking and non-blocking operations
    if(BLOCKING_TEST & ((passCount & 2) == 0) || !NON_BLOCKING_TEST)) {
        mcap PageSize send(endpoint[LOCAL_DOMAIN][LOCAL_NODE][1 - 1], endpoint[0][1][1 - 1],
        if(mcap_status != MCAPI_SUCCESS) {
            printf("Error on blocking send: %s\n", mcap_display_status(mcap_status, sta
            sendError++;
        }
        mcap PageSize recv(endpoint[LOCAL_DOMAIN][LOCAL_NODE][1 - 1], (void*) (void*)recv_data
        if(mcap_status != MCAPI_SUCCESS) {
            printf("Error on blocking recv: %s\n", mcap_display_status(mcap_status, sta
            receiveError++;
        }
    }
```
Portability Example - Tools

Portability of the core and repository configurations across different operating systems and processor types.
Portability Example - Tools

Related Video:
polycoresoftware.com/solutions/video
Conclusions

Market Requirements ➤ Increasing Complexity

Solutions

• Multicore SoC
• Programming Model, Standardized
• Unified Tools
  • Application to Multicore SoC Abstraction
  • Access to Acceleration

Portability
Thank You

www.ti.com
www.PolyCoreSoftware.com
www.multicore-association.org