The DSP/BIOS Bridge

Víctor Manuel Jáquez Leal

Igalia

06 February 2010

Parallel computing

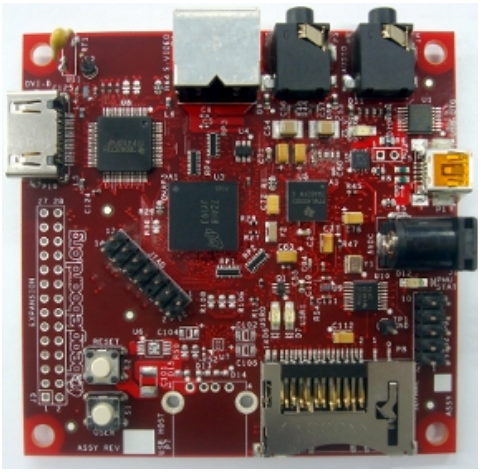
- Serial computing is now dead.
  - Parallel computing (which started more than 40 years ago) is a revolution that is now upon us
- Programming for serial computing is already difficult
  - Programming for parallel computing will only exacerbate this difficulty
  - For parallelism to succeed it must produce better performance, efficiency and reliability

OMAP 3530/20

- 720 MHz ARM Cortex A8
- 520 MHz TMS320C64x+ DSP
- POWERVR SGX Graphics Accelerator

Devices and boards using OMAP3

- There are a lot!



The C64x+ DSP

- Digital Signal Processor
  - Specialized microprocessor
  - For fast execution of digital signal processing
  - Low power consumption
- Digital Signal Processing
  - Measurement and filtering of continuous real-world analog signals
  - Audio, video, speech, are examples of those signals

DSP-GPP parallel computing

- Features to control the DSP
- Mechanisms to communicate with DSP
- Enabling parallel processing for multimedia acceleration

Available drivers

- dsp-gateway
  - Developed by Nokia for the Maemo Internet Tablets
  - It works on OMAP1 and OMAP2
  - It's production ready
  - It's used on the Nokia N800 and N810
  - It follows Linux standards and it's close to upstream acceptance
  - There's code for OMAP3 but it hasn't been thoroughly tested

Available drivers

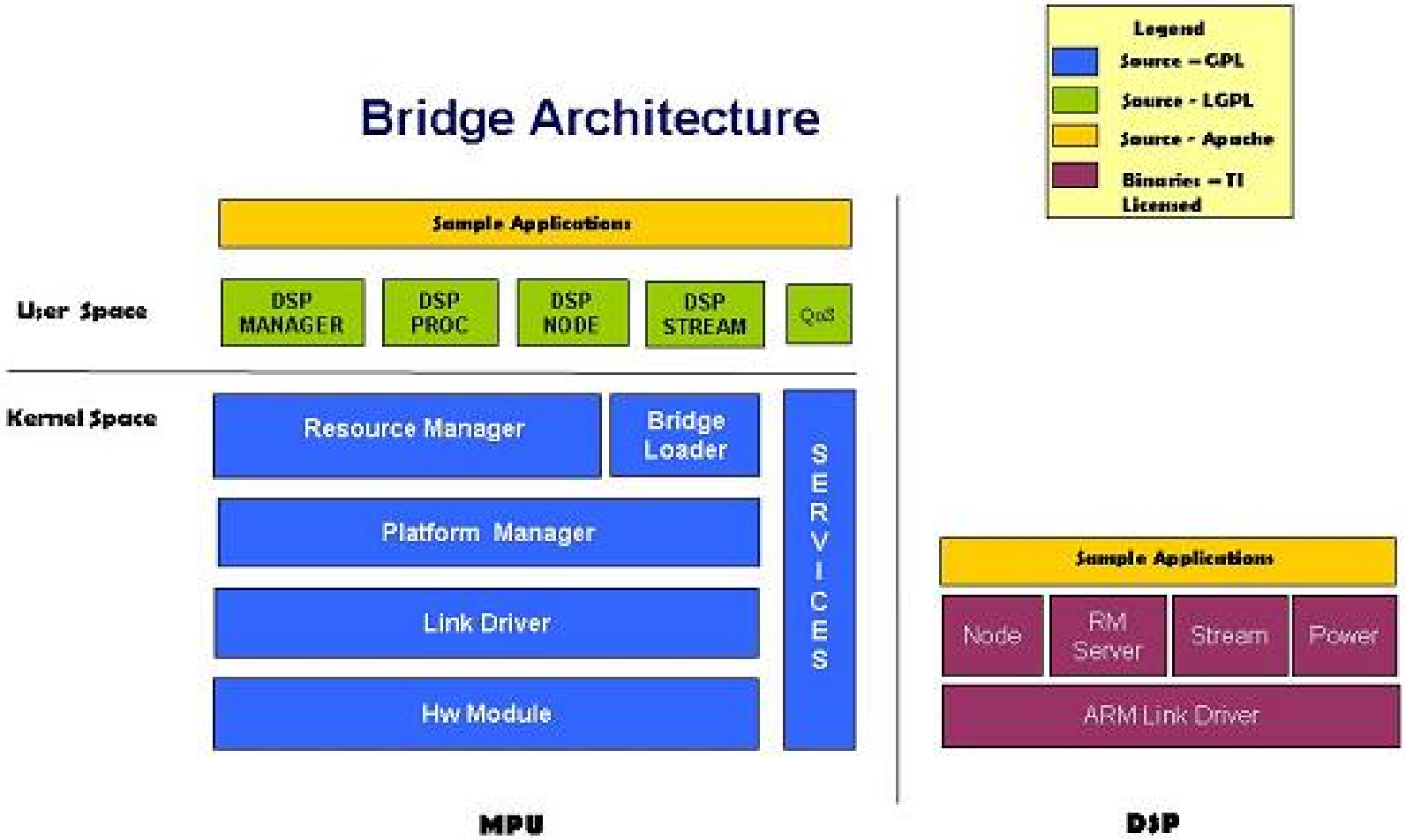
- dsp-bridge
  - Originally developed by TI
  - It still doesn't meet Linux standards although there has been a lot of progress
  - Only the ARM side is available as open source; the DSP side is completely closed

Available drivers

- dsp-link
  - A slimmer version of the dsp-bridge
  - Also developed by TI
  - It supports a wide variety of devices (DaVinci, OMAP2, OMAP3, etc)
  - The kernel driver doesn't meet the Linux kernel coding conventions
  - The sources haven't been submitted for review, and it is not currently planned to be merged into upstream kernels

DSP/BIOS Bridge

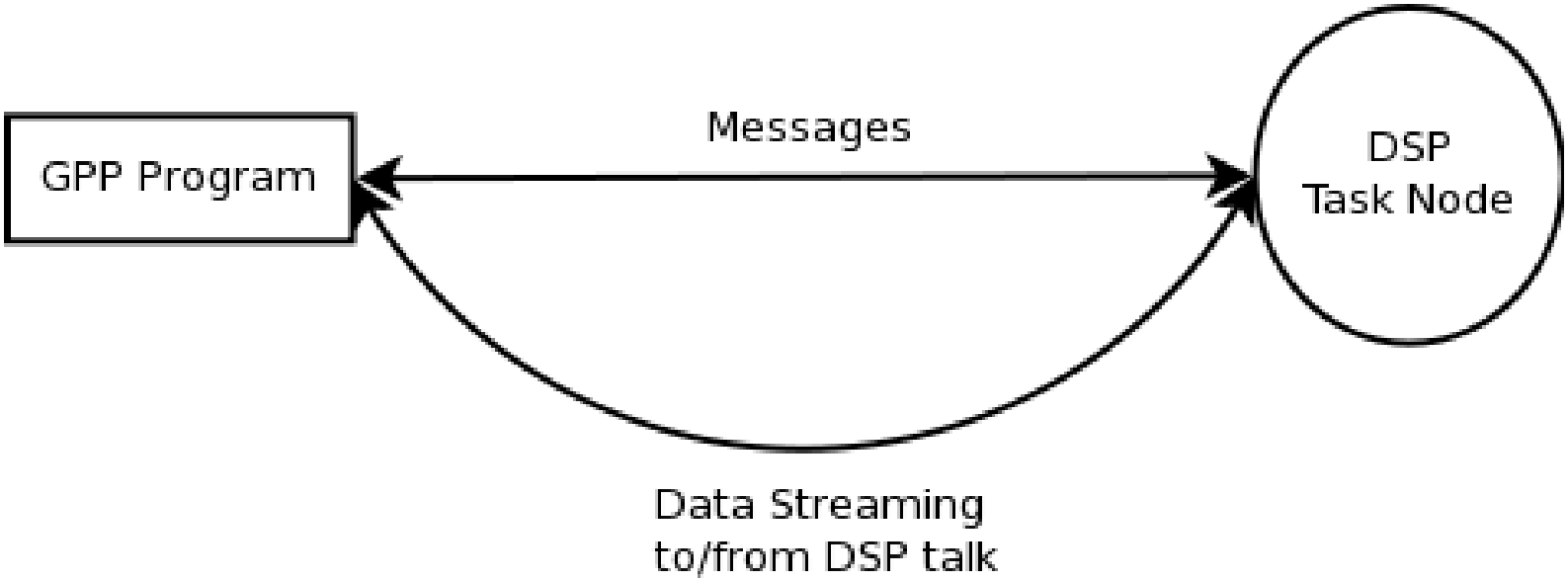
- General Architecture



Architecture

- It is designed for one GPP and one or more attached DSPs
  - The GPP is considered the master or "host" processor
  - The attached DSPs are processing resources that can be used by applications running on the GPP

Architecture



Architecture

- The Bridge supplies a link between a GPP program and a DSP task
- The communication link is partitioned into two types:
  - Messaging (short, fixed-length packets): For passing control and status information
  - Data streaming (multiple, large buffers): for streaming real-time data
- Each sub-link operates independently
- A GPP client can specify what inputs and outputs a DSP task uses

GPP Sopftware Architecture

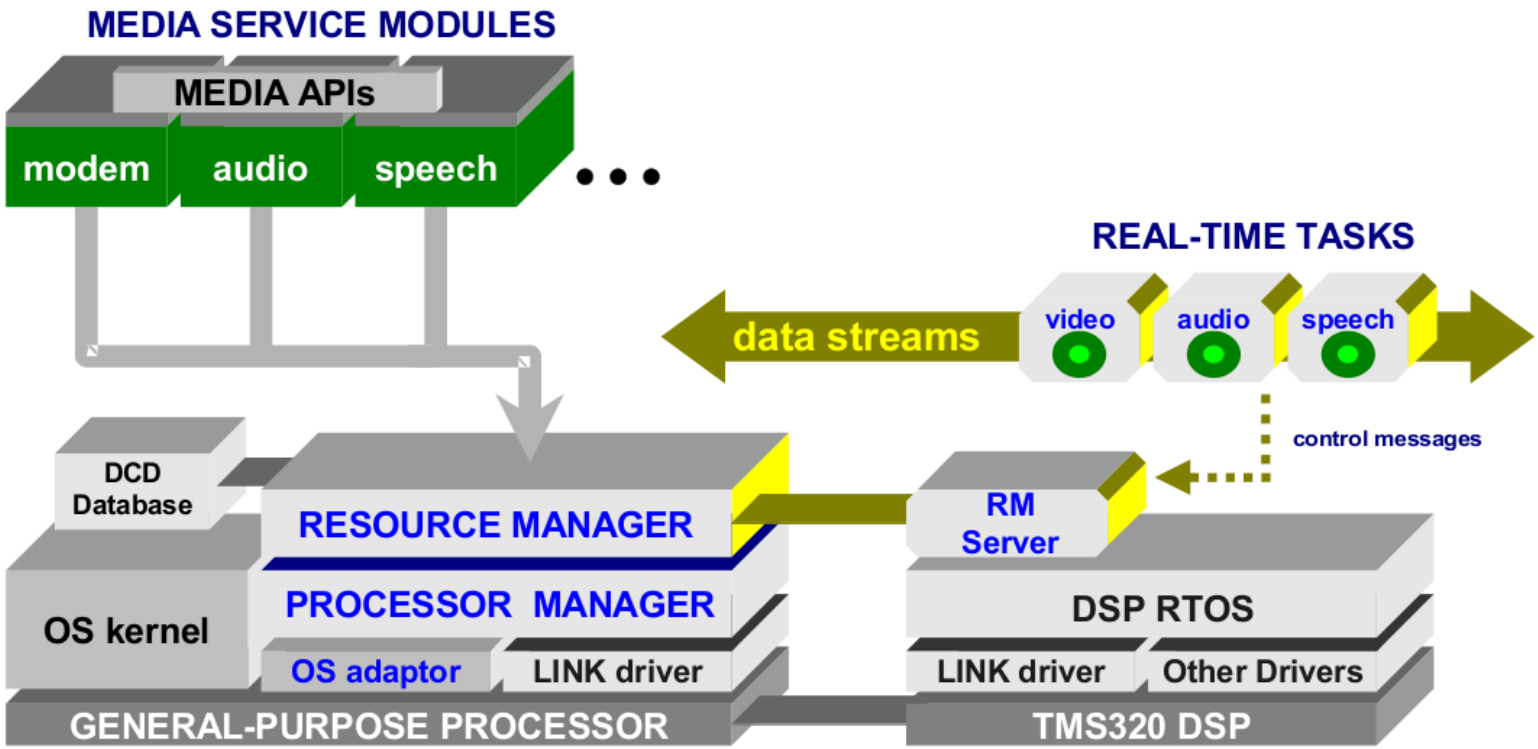
- The GPP OS see the DSP just as another peripheral device

```
root@beagleboard:~# ls -la /dev/DspBridge
crw-rw---- 1 root root 251, 0 Jan 1 2000 /dev/DspBridge
```

DSP Software Architecture

- From the DSP/BIOS perspective, the bridge provides
  - A device-independent streaming I/O (STRM) interface
  - A messaging interface (NODE)
  - A Resource Manager (RM) Server
    - The task environment is established by the RM Server

Components


MEDIA SERVICE MODULES
MEDIA APIs
modem
audio
speech
REAL-TIME TASKS
data streams
video
audio
speech
control messages
DCD
Database
RESOURCE MANAGER
RM
Server
OS kernel
PROCESSOR MANAGER
DSP RTOS
OS adaptor
LINK driver
LINK driver
Other Drivers
GENERAL-PURPOSE PROCESSOR
TMS320 DSP

Components (GPP)

- Resource Manager
  - Dynamically instantiating DSP resources
  - Monitoring DSP resources
  - Dynamically loading DSP code as needed
  - Implementing policies for managing DSP resources
- Platform Manager
  - Statically loading a base code image for the DSP
  - Starting and stopping the DSP
  - Implementing data streaming

Components (GPP)

- OS adaptation layer
- DSP link driver for low level communication
- A dynamic configuration database (DCD) stores configuration information

Components (DSP)

- DSP/BIOS communicates with the GPP via the link driver
- On top of the DSP/BIOS sits the Resource Manager (RM) Server
  - Dynamically create, execute and destroy DSP processing nodes under Resource Manager control
  - Routing messages between the GPP and individual nodes
  - Altering task priorities
  - Responding to Resource Manager configuration commands
  - Status queries

Components (DSP)

- DSP task nodes
  - They are separate execution threads running on the DSP
  - They implement control or signal processing algorithms
  - They communicate with one another, and with the GPP
    - via short fixed length messages and/or device-independent stream I/O.

GPP Side interface

- Manager
  - Used to obtain DSP processor and manipulate node configuration information
- Processor
  - Used to manipulate DSP processor objects, which represent particular DSP subsystems linked to the GPP
  - Processor objects are used to create, execute and delete nodes on a particular DSP subsystem
  - As DSP/BIOS Bridge clients make processor API calls, the corresponding DSP processor will transition between a set of pre-defined states.

GPP Side interface

- Node
  - Used to manipulate node objects, which represent control and signal processing elements running on a particular DSP
- Stream
  - Used to manipulate stream objects, which represent logical channels for streaming data between the GPP and nodes on a particular DSP

Sequence for nodes controlling

- Load the device driver

```
root@beagleboard:~# lsmod
Module                  Size  Used by
dspbridge                729  0
bridgedriver          187569  1
```

- Load a base image to the DSP

```
~# cat /etc/modprobe.d/bridgedriver.conf
options bridgedriver base_img=/lib/dsp/baseimage.dof
```

Sequence for nodes controlling

- Open a handle to the DSP/BIOS Bridge device

```
dsp_handle = dsp_open();

if (dsp_handle < 0) {
  pr_err("failed to open DSP");
  return -1;
}
```

- Reserve GPP-side resources for controlling a particular DSP

```
if (!dsp_attach(dsp_handle, 0, NULL, &proc)) {
   pr_err("dsp attach failed");
   ret = -1;
   goto leave;
}
```

Sequence for nodes controlling

- Allocate DSP node for the selected processor

```
const dsp_uuid_t uuid =
  { 0x12a3c3c1, 0xd015, 0x11d4, 0x9f, 0x69,
    { 0x00, 0xc0, 0x4f, 0x3a, 0x59, 0xae } };

if (!dsp_node_allocate(dsp_handle, proc, &uuid,
                       NULL, NULL, &node)) {
   pr_err("dsp node allocate failed");
   return NULL;
}
```

Sequence for nodes controlling

- Create the node on the DSP

```
if (!dsp_node_create(dsp_handle, node)) {
    pr_err("dsp node create failed");
    return NULL;
}
```

Sequence for nodes controlling

- Launch the task node into their execute phase

```
if (!dsp_node_run(dsp_handle, node)) {
    pr_err("dsp node run failed");
    return false;
}
```

  - Once the task is running, the GPP client can stream data buffers to/from the task as well as exchange short messages with the task

Sequence for nodes controlling

- Stream data to/from DSP tasks
  - The GPP client then allocates data buffers for the stream
    - If the buffer are already pre-allocated, the GPP client can prepare the buffers for the stream
  - Once allocated and prepared
    - They can be used to submit buffers to a stream
    - Submitting a data buffer to a stream will not block GPP thread execution.
    - They can request a buffer back from a stream
    - Requesting a buffer back from the stream may cause the GPP thread to block

Sequence for nodes controlling

- Exchange messages with DSP nodes

```
if (!dsp_send_message(dsp_handle,node,1,0,0)) {
    pr_err("dsp node put message failed");
    continue;
}
```

```
if (dsp_node_get_message(dsp_handle,node,&msg,0))
    printf("Ping: Id %d Msg %d Mem %d\n",
           msg.cmd, msg.arg_1, msg.arg_2);
```

Sequence for nodes controlling

- Terminate DSP nodes

```
if (!dsp_node_terminate (dsp_handle, node,
                         &exit_status)) {
    pr_err("dsp node terminate failed: %lx",
           exit_status);
    return false;
}
```

Sequence for nodes controlling

- Delete DSP nodes

```
if (!dsp_node_free(dsp_handle, node)) {
    pr_err("dsp node free failed");
    return false;
}
```

The kernel

- DSP/BIOS Bridge driver is not in Linus' branch yet
- Neither in linux-omap's main branch
  - http://git.kernel.org/?p=linux/kernel/git/tmlind/linux-omap-2.6.git;a=shortlog;h=refs/heads/dspbridge
  - http://dev.omapzoom.org/?p=tidspbridge/kernel-dspbridge.git;a=shortlog;h=refs/heads/dspbridge
  - http://gitorious.org/~felipec/linux-omap/felipec

The GPP libraries

- TI dbapi
  - http://dev.omapzoom.org/?p=tidspbridge/userspace-dspbridge.git;a=summary
- dsp_bridge
  - http://github.com/felipec/gst-dsp

Applications

- Samples
  - http://github.com/felipec/dsp-dummy
  - http://gitorious.org/vjaquez-beagleboard/dsp-samples
- Applications
  - http://maemo.gitorious.org/maemo-multimedia/dsp-tools
  - http://github.com/felipec/gst-dsp

Socket Nodes

- Samples
  - http://dev.omapzoom.org/?p=tidspbridge/userspace-dspbridge.git;a=summary
- Multimedia
  - Part of OpenMAX
  - https://gforge.ti.com/gf/project/openmax/frs/
  - http://code.entropywave.com/git?p=leonora.git;a=tree

C64x+ toolchain

- Free as beer
- Compiler
  - https://www-a.ti.com/downloads/sds_support/TICodegenerationTools/download.htm
- DSP/BIOS (libraries)
  - http://software-dl.ti.com/dsps/dsps_registered_sw/sdo_sb/targetcontent/bios/index.htn
- doffbuild tools
  - Part of the userspace-dspbridge package

All together

- Marmita
  - OE recipes overlay
  - It is a work in progress
  - Only tested in the Beagleboard (rev B6)
  - Minimal image (10Mb)
  - http://gitorious.org/vjaquez-beagleboard/marmita

Marmita

- It's based on Angstrom distribution
- Provides recipes for
  - felipec's kernel 2.6.32
    - DSS2
    - dspbridge
  - gst-dsp
  - dsp-tools
  - dsp-samples
  - libbridge (dspbridge API)
  - libomx-ti

OMAP4

- DSP/BIOS Bridge will be deprecated :(
  - syslink is the new thing
    - ARM M3 1GHz dual core (Ducati)
    - DSP TMS320C64x (Tesla)
    - ARM A9 1GHz dual core

http://dev.omapzoom.org/?p=tisyslink/kernel-syslink.git;a=summary

The trend

- More cores
- More processing units
- More heterogeneity
- MORE COMPLEXITY

Thank you

- Questions?