SMP and Networking support on NuttX / LC823450

Masayuki.Ishikawa@sony.com
Senior Software Engineer
Sony Home Entertainment & Sound Products Inc.
Agenda

- Development history (NuttX-based products)
- SMP (Symmetric Multiprocessing) related status
- Networking related status
- Demo videos
Development history*(NuttX-based products)

- Oct 2013 -
  - Ported NuttX to LC823425 (ARM7)
- Apr 2014 -
  - Ported bluetooth stack to NuttX + QEMU
- Jul 2014 -
  - Ported NuttX to LC823450 (Cortex-M3) FPGA
- Jan 2015 -
  - Migrated to LC823450-ES board
- Sep 2015 -
  - Released the first NuttX-based audio products.
- Oct 2016 -
  - Talked at Arm TechCon 2016, ELC NA 2017 ** and OpenIoT NA 2018

*https://www.youtube.com/watch?v=TjuzH6JthxQ ** https://www.youtube.com/watch?v=T8fLjWyl5nI
About NuttX and why we chose it

- POSIX and libc are supported
  - Can reuse existing software
  - Can reduce training costs
- ELF* is supported
  - Can divide into small apps
- Driver framework is supported
  - Helps us implement drivers
- Has Linux-like configuration system
  - Helps us develop multiple products
- Many MCUs and boards are supported
  - Helps us port NuttX to new MCU
- Provided with BSD license

* ELF = Executable and Linking Format

From http://www.nuttx.org/
LC823450 Features

- ARM dual Cortex-M3
- 32bit fixed point, dual-MAC original DSP
- Internal SRAM (1656KB) for ARM and DSP
- I2S I/F with 16/24/32bit, MAX 192kHz (2chx2)
- Hard wired audio functions
  - MP3 encoder and decoder, EQ (6-band equalizer), etc.
- Integrated analog functions
  - Low-power Class D HP amplifier, system PLL
  - Dedicated audio PLL, ADC
- Various interfaces
  - USB2.0 HS device / host (not OTG), eMMC, SD card, SPI, I2C, etc.
- ARM and DSP clock max frequency
  - 160MHz at 1.2V
  - 100MHz at 1.0V

From http://www.onsemi.com/PowerSolutions/product.do?id=LC823450
AMP vs SMP in general *

- Asymmetric multiprocessing (AMP)
  - A separate OS, or a separate copy of the same OS, manages each core.
  - Provides an execution environment similar to that of uniprocessor system, allowing simple migration of legacy code. Also allows developers to manage each core independently.

- Symmetric multiprocessing (SMP)
  - A single OS manages all processor cores simultaneously. The OS can dynamically schedule any process on any core.
  - Provides greater scalability and parallelism than AMP, along with simpler shared resource management

* http://www.embeddedintel.com/special_features.php?article=189
Why SMP with LC823450?

- **Motivation**
  - Run existing applications in SMP mode
  - Establish knowledge on debugging
  - Confirm *performance penalty*
  - Confirm power consumption
  - Very challenging theme (because NuttX is not just a scheduler)

- **Other reasons…**
  - The architecture is much simpler than quad Cortex-A9.
  - Suitable system to understand SMP kernel.

* Note that LC823450 does not have CPU cache but has multiple SRAM segments
### Introduction to the NuttX SMP kernel

- **Minimum changes to non-SMP kernel**
  - CONFIG_SMP is introduced.
  - Main changes are done in the scheduler.

- **Newly introduced**
  - Spinlock to protect shared resources
  - Critical section APIs to replace with local interrupt control APIs.
  - pthread_setaffinity_np(), sched_setaffinity() are supported

- **H/W interrupts except for inter-CPU interrupts are assumed to be handled at CPU0**
  - To prevent deadlocks

---

**SMP**

**Definition**

According to Wikipedia: "Symmetric multiprocessing (SMP) involves a symmetric multiprocessor system hardware and software architecture where two or more identical processors connect to a single, shared main memory, have full access to all I/O devices, and are controlled by a single operating system instance that treats all processors equally, reserving none for special purposes. Most multiprocessor systems today use an SMP architecture. In the case of multi-core processors, the SMP architecture applies to the cores, treating them as separate processors."

*SMP systems are tightly coupled multiprocessor systems with a pool of homogenous processors running independently, each processor executing..."
NuttX SMP : available boards

- NXP (Freescale) i.MX6 Quad Sabre
  - Quad Arm Cortex-A9
  - SMP kernel can run on QEMU *
- Espressif Systems ESP32
  - Dual Tensilica LX6 *

- Microchip (Atmel) SAM4CMP-DB
  - Arm Cortex-M4 w/MPU + Cortex-M4F *
- ON Semiconductor LC823450XGEVK
  - Dual Arm Cortex-M3
  - Approx. $46 **

*ostest still has some issues.  **http://www.components-center.com/product/ON-Semiconductor/LC823450XGEVK.html
Running SMP kernel : SAM4CMP-DB

- Cortex-M4 /w MPU + Cortex-M4F
  - Not symmetric, but if both CPU does not use MPU nor FPU, it should be OK.
  - Each CPU has local SRAM which can be accessed via bus bridge from another CPU.

- Bus bridge issue *
  - “ostest” crashes due to CPU lockup or hardfault
  - It’s difficult to assure memory access just by memory barrier operations.
  - Dummy memory read/write might resolve this issue, but we still can not find the correct way.
  - We asked this issues to Atmel before, but no response received yet.

* I don’t think this board can perfectly work in SMP mode
Running SMP kernel : LC823450XGEVK

- Port existing drivers to the latest NuttX
  - UART, Timer, GPIO, DMA, I2C, SPI, LCD
  - eMMC (including boot), SD, USB, ADC, …

- Implement SMP related code
  - lc823450_cpuidlestack.c, lc823450_cpuindex.c
  - lc823450_cpupause.c, lc823450_cpustart.c,
    lc823450_testset.c (NOTE: H/W Mutex is used instead of Idex, strex)

- Performance improvement
  - Introduced spin_lock_irqsave(), spin_unlock_irqstore()
  - Applied APIs inside the driver code.
  - Up to 20% performance improvement achieved
Tracing SMP kernel

- What events can be traced
  - SMP specific (inter-CPU communication)
    - CPU_PAUSE, CPU_PAUSED, CPU_RESUMED
  - SMP/non-SMP common
    - SUSPEND, RESUME (context switch)
    - PREEMPT_LOCK, PREEMPT_UNLOCK

- Tools
  - Use gdb macro to dump the trace buffer
  - Use “noteinfo” to analyze the dump file
OpenOCD for lc823450-smp*

- **Implementation**
  - Understand how Cortex-A SMP support works in OpenOCD
  - Modify several files (target/cortex_m.c …) to support Cortex-M in SMP mode
  - Specify APSEL (Access Port Selection) when accessing to each core in LC823450
  - Modify tcl/target/lc823450.cfg to support multiple debug access ports and targets.
  - Modify rtos/nuttx.c to show SMP related tasklists

*Code is NOT merged yet.
Debugging an SMP application

- Modify hello_main.c
  - Assign the current task to CPU1 (not CPU0)
  - Print CPU index.
- Add a break point at printf()
- Run “hello” on the nsh
- Break point hits on CPU1
- Check the trace log
Enhance DVFS* for SMP

- Need to handle both CPUs
  - 1. If at least one CPU is active, apply the active mode clock.
  - 2. If both CPUs are idle (i.e., WFI), apply the idle mode clock.

- Calculate CPU idle time on both CPUs
  - 3. If at least one CPU falls below the lower threshold (e.g., 20% idle), go to the higher clock mode.
  - 4. If both CPUs exceed the higher threshold (e.g., 70% idle), go to the lower clock mode.

* See also: https://www.youtube.com/watch?v=T8fLjWyI5nI
CPU activity examples* (1/2)

Usage: taskset mask command [args]

mask=1 assigns CPU0, mask=2 assigns CPU1, mask=3 assigns CPU0 or CPU1

* CH1=Cortex-M3 #0, CH2=Cortex-M3 #1
CPU activity examples (2/2)

- **Background**
  - LC823450 has 3 SDIO controllers.
  - eMMC uses CH0, uSD uses CH1.
  - Accessing different channels will be faster than accessing the same channel.

- (1) Two md5 for the same channel
  - Concurrent access is impossible.
  - Results: 11.0 sec & 11.0 sec (file size=6.6MB)
  - NOTE: 5.9 sec (eMMC single access)

- (2) Two md5 for different channels
  - Concurrent access is possible.
  - Results: 7.8 sec & 7.9 sec (file size=6.6MB)
  - NOTE: 6.2 sec (uSD single access)

* uSD: SanDisk 16GB (SDSDQUP-016G-J35A)
Power consumption comparison

- nxplayer with local playback
  - WAV file 44.1kHz/16bit/2ch on eMMC
  - Vdd1=1.0V *
  - CPU clock = 40MHz (active), 6MHz(idle)

- Power consumption** @Vdd1
  - SMP : 6.0mA (idle=3.6mA)
  - non-SMP : 4.4mA (idle=3.5mA)

Performance penalty in SMP mode is outstanding (i.e. bus conflicts and scheduling overhead). However, more optimization would be possible.

*Power consumption of the logic part (i.e. Cortex-M3, SRAM, DMA, I2S, …) inside the MCU
**Audio paths are need to be changed as of OpenIoT NA 2018
Networking with LC823450XGEVK

- Motivation
  - Confirm NuttX network stack feasibility
    - IPv4, IPv6, ICMP, UDP, TCP, ...
  - Run the network stack with minimum efforts.
    (We already have an USB driver for LC823450)
  - Audio streaming (PCM and MP3)
  - Run the network stack in SMP mode
  - Do various tests via telnet
NuttX networking features

- Ethernet and IEEE 802.11 Full MAC
- 6LoWPAN for radio network drivers (IEEE 802.15.4 MAC)
- USB RNDIS (since 7.23), CDC-ECM (since 7.26)
- SLIP, TUN/PPP, local loopback devices
- IPv4, IPv6, TCP, UDP, ARP, ICMP, ICMPv6, IGMPv2
- IP forwarding
- BSD compatible socket layer
- DNS name resolution / NetDB
- User socket (listen/accept are supported in 7.26)
- Bluetooth socket
PCM audio streaming via RNDIS

- Fix RNDIS driver for NuttX
  - Fix data corruption
  - Add USB high speed mode support
- Receive window control has been added
  - Need more improvement due to packet drop
- Modify nxplayer to support HTTP streaming
  - Currently only WAV format is supported.
- Still testing with SMP kernel
  - In various conditions (clock speed, network traffic, etc)
PCM audio streaming example

- ‘ps’ command results shows
  - Dual CPUs are running
  - telnet daemon is running
  - one telnet session is running
  - nxplayer is running

- ‘ifconfig’ command results shows
  - private address has been assigned via DHCP
  - TCP/UDP traffic (NOTE: some TCP packets are dropped due to iob starvation, so TCP flow control should be improved)
Network traffic and CPU activity examples

Network traffic when PCM audio (44.1k/16bit/2ch) streaming is working

(1) CPU clock : 160MHz (fixed)

(2) CPU clock : 160/80/40MHz (auto)
MP3 audio streaming via Bluetooth

- Port the BTstack* by Bluekitchen to NuttX
  - Based on posix-h4** with H/W flow control
  - UART speed : 921600 baud
  - Tested with iOS/Android/macOS/OpenWrt
  - Free for non-commercial use

- Add TAP mode to the NuttX tun driver
  - TAP mode is used for network bridge
  - NOTE: TUN mode is used for network routing

- Add H/W MP3 decoder to lc823450_i2s.c

- HCI_RESET issue in SMP mode
  - CSR’s mode change with HCI_RESET is tricky
  - Still unstable in SMP mode

* https://bluekitchen-gmbh.com/
** We can use posix-h5 (3-wire protocol) as well. However, it has performance drawbacks.
Running the BTstack on NuttX

- tinyplay
- renew (dhcp client)
- TCP
- UDP
- IPv4
- ICMP
- ARP
- tun (bnep0)
- UART
- CSR8811
- BT PAN profile

Profiles:
- GATT
- SPP
- HSP
- HFP
- MFI
- SDAP
- PAN
- GAP

Protocols:
- SMP
- ATT
- RFCOMM
- SDP
- BNEP
- L2CAP LE
- L2CAP
- HCl

Transport:
- H1 UART
- eHCI UART
- ... H2 USB

BTSTACK

OpenWrt PAN-NAP* role
DHCP server

WZR-HP-G300NH

*PAN: Personal Area Network
*BNEP: Bluetooth Network Encapsulation Protocol
*NAP: Network Access Point
BTstack log example

H4 device: /dev/ttyS1
[2019-06-27 12:12:42.280] LOG -- hci.c.3797: BTSTACK_EVENT_STATE 1
[2019-06-27 12:12:42.810] LOG -- hci.c.1878: Manufacturer: 0x000a
Local version information:
- HCI Version 0x0006
- HCI Revision 0x2031

[2019-06-27 12:12:57.070] LOG -- b neurop.c.79: BNEP EVENT_CHANNEL_OPENED status 0x00 bd addr: 00:18:DC:06:86:59 source UUID 0x1115 dest UUID: 0x1116,
BNEP connection open succeeded to 00:18:DC:06:86:59 source UUID 0x1115 dest UUID: 0x1116,
Network Interface b neurop0 activated
MP3 streaming via Bluetooth

Realtime Traffic

- **bne0**
- **br-lan**
- **eth0**
- **eth0.1**
- **eth1**
- **wlan0**

**Inbound:** 4.59 kbit/s (0.57 kB/s)
**Average:** 4.17 kbit/s (0.52 kB/s)

**Outbound:** 181.04 kbit/s (22.63 kB/s)
**Average:** 157 kbit/s (19.63 kB/s)

Longitudinal graph showing network traffic over time.
Demo videos

- CPU activity examples (busyloop, md5)
- HTTP PCM audio streaming via RNDIS
Any Questions?