BKK19-301 SMP Development on ARM Board

Satish Kumar G
Index

1. Introduction to SMP Functionality
2. SMP Functions call flow
3. Debug SMP Call Flow
4. Usage
1. Introduction to SMP Functionality

1. **Symmetric Multi-Processing (SMP)** is a software architecture that dynamically determines the roles of individual cores.
2. Each core in the cluster has the same view of memory and of shared hardware.
3. Any application, process, or task can run on any core and the operating system scheduler can dynamically migrate tasks between cores to achieve optimal system load.
2. SMP Functions Call Flow

1. SMP supported functions call flow with Zynq Zc702 board.
   - **Board used:** ZEDBoard (Xilinx Zynq Zc702, ARM Cortex A9 Dual Core)
   - **Kernel version:** linux-xlnx (2014-03)

2. SMP call flow involves
   - 2.1 Core Kernel Functions (Platform independent code)
   - 2.2 Arch. Dependent code
   - 2.3 Board SMP Code (platsmp.c of linux-xlnx)
2. SMP Functions Call Flow

**SMP Identification:**
1. The code is running on CPU0 or CPU1 is identified by reading co-processor 15(CP15).
   This register is named as **MPIDR (Multiprocessor Affinity Register)**.

2. The primary core finishes the boot loader and immediately starts running Linux, while the secondary core waits in the boot loader for a **jumping address**.

3. The primary CPU which is running Linux is responsible to release the secondary CPU by **writing the jump address and sending an interrupt or event**.
   (This code is part of arch/arm/mach-****/platsmp.c)
2. SMP Functions Call Flow

SMP Functions relates to SoC:

```c
struct smp_operations {
    void (*smp_init_cpus)(void);
    void (*smp_prepare_cpus)(unsigned int max_cpus);
    void (*smp_secondary_init)(unsigned int cpu);
    int (*smp_boot_secondary)(unsigned int cpu,
                              struct task_struct *idle);
    [...]  
};
```

Note: Only `smp_boot_secondary()` is mandatory, others are optional and depend on the need of the SoCs.
2. SMP Functions Call Flow

1. **smp_init_cpus()** :
   1.1 This function is called early in boot process using `setup_arch()`.
   1.2 Sets the set of possible CPUs using `cpu_possible()`.

2. **smp_prepare_cpus()** :
   2.1 Enables Coherency.
   2.2 Initializes `cpu_possible` map.
   2.3 Prepares resources (power, ram, clock).
   2.4 Called early during boot process (Before init calls & after `setup_arch()`).
2. SMP Functions Call Flow

3. \texttt{smp\_secondary\_init()}:  
   3.1 Supposed to “perform platform specific initialization of the specified CPU”  
   3.2 Called from \texttt{secondary\_start\_kernel()} on the CPU which has just been started.

4. \texttt{smp\_boot\_secondary()}:  
   4.1 Actually boots a secondary CPU identified by the CPU number given in parameter.  
   4.2 Called from \texttt{cpu\_up()} on the booting CPU

Reference: https://elinux.org/images/0/00/Clement-smp-bring-up-on-arm-soc.pdf
2. SMP Functions Call Flow

1. init/main.c: start_kernel()
2. kernel/time/tick-common.c : tick_init()
3. init/main.c: boot_cpu_init()
4. mm/highmem: page_address_init()
5. arch/arm/kernel/setup.c : setup_arch()
6. arch/arm/kernel/smp.c: smp_init_cpus()
7. arch/arm/kernel/smp.c: cpu_init()
8. arch/arm/kernel/smp.c: smp_prepare_boot_cpu()
9. kernel/sched/core.c: sched_init()
10. mm/page_alloc.c : __build_all_zonelists()
11. init/main.c: parse_early_param()
12. init/main.c: parse_early_options()
13. arch/arm/kernel/irq.c: init_IRQ()
14. arch/arm/kernel/traps.c: trap_init()
15. kernel/softirq.c: softirq_init()
16. arch/arm/kernel/time.c: time_init()
17. drivers/tty/tty_io.c: console_init()
18. arch/arm/mm/init.c: mem_init()
19. mm/slab.c: kmem_cache_init()
20. init/calibrate.c: calibrate_delay()
2. SMP Functions Call Flow

21. fs/proc/root.c: proc_root_init()
22. init/main.c: rest_init()
23. kernel/sched/core.c: __schedule()
24. arch/arm/mach-zynq/platsmp.c: zynq_smp_prepare_cpus()
25. arch/arm/kernel/smp.c: smp_store_cpu_info()
26. arch/arm/kernel/smp_scu.c: scu_enable()
27. arch/arm/mach-zynq/platsmp.c: zynq_cpun_start()
28. arch/arm/kernel/smp.c: secondary_start_kernel()
29. arch/arm/kernel/setup.c: cpu_init()
   Same as Function no. 7
30. arch/arm/mach-zynq/platsmp.c: zynq_secondary_init()
31. init/main.c: boot_cpu_init()
   Same as Function no:3
32. init/calibrate.c: calibrate_delay()
   Same as Function no:20
33. arch/arm/kernel/smp.c: smp_store_cpu_info()
   Same as Function no:25
2. SMP Functions Call Flow

34. arch/arm/kernel/smp.c:
   __cpu_up()
   Boot Ever offline CPU
35. arch/arm/kernel/process.c:
   arch_cpu_idle()
   Same as Function no:23
36. kernel/fork.c:
   fork_idle() ->
   Create new Idle process for each CPU
37. arch/arm/mach-zynq/platsmp.c
   zynq_boot_secondary()
38. arch/arm/kernel/smp.c:
   smp_cpus_done() ->
   Display no. of CPUs
39. init/main.c:
   do_basic_setup()

Note: Bold is Platform dependent functions
& remaining is Platform Independent functions.
3. Debug SMP call flow

Adding Debugging statements using ‘printk’ to get the function flow, starting from kernel image decompression & reaches to SMP count (no.of CPUS) present in the Board.
3. Debug SMP call flow

3694108 bytes read

## Starting application at 0x00000000 ...

Uncompressing Linux... done, booting the kernel.

- 0.000000 Satish:Kernel init entry: Function 1
- 0.000000 Booting Linux on physical CPU 0
- 0.000000 Satish: Init Kernel Tick Init: Function 2
- 0.000000 Satish:Activate First CPU :Function 3
- 0.000000 satish: cpu = 0 Satish:Function 3
- 0.000000 Satish:Memory Subsystem:Function 4
- 0.000000 Linux version 3.6.6-digilent-13.01-00002-gaa40afb (satish@satish-laptop) (gcc version 4.8.3 20140320 (prerelease) (Source)
- 0.000000 Satish:Setup arch: Memory port&I/O access:Function 5
- 0.000000 CPU: ARMv7 Processor [413fc090] revision 0 (ARMv7), cr=18c5387d
- 0.000000 CPU: PIPT / VIPT nonallasing data cache, VIPT allasing instruction cache

- 0.000000 Satish:CPU_Init:Function 7
- 0.000000 Satish:CPU_Init for Secondary CPU:Function 29
- 0.000000 Machine: Xilinx Zynq Platform, model: Xilinx Zynq ZED
- 0.000000 Satish:Parse Kernel args:Function 11
- 0.000000 Satish:Parse early args:Function 12
- 0.000000 Memory policy: ECC disabled, Data cache writealloc
- 0.000000 Satish:CPU Map:Function 6
- 0.000000 1297 Satish is in scu_get_core_count
- 0.000000 Satish: cpucore = 2 Satish:Function 6
- 0.000000 Satish:setup_per_cpu_area:Function 8
- 0.000000 PERCPU: Embedded 7 pages/cpu @c1408000 s6976 r8192 d13504 u32768
- 0.000000 Satish:Access storage data:Function 8
- 0.000000 Satish:Initiate memory Zones:DMA/Normal/Highmeme :Function 10

- 0.000000 Built 1 zonelists in Zone order, mobility grouping on. Total pages: 130048
4. Debug SMP call flow

0.000000] Satish:CPU_Init for Secondary CPU:Function 29
0.000000] Machine: Xilinx Zynq Platform, model: Xilinx Zynq ZED
0.000000] Satish:Parse Kernel args:Function 11
0.000000] Satish:Parse early args:Function 12
0.000000] Memory policy: ECC disabled, Data cache writealloc
0.000000] Satish:CPU Map:Function 6
0.000000] 1297 Satish is in scu_get_core_count
0.000000] Satish: cpucores = 2 Satish:Function 6
0.000000] Satish: setup_per_cpu_area:Function 8
0.000000] PERCPU: Embedded 7 pages/cpu @c1408000 s6976 r8192 d13504 u32768
0.000000] Satish: Access storage data:Function 8
0.000000] Satish: Initiate memory Zones:DMA/Normal/Highmeme:Function 10
0.000000] Build 1 zonelists in Zone order, mobility grouping on. Total pages: 130048
0.000000] Kernel command line: console=ttyPS0,115200 root=/dev/ram rw initrd=0x800000,8M earlyprintk rootfstype=ext4 rootwait devt0
0.000000] Satish: Parse Kernel args:Function 11
0.000000] PID hash table entries: 2048 (order: 1, 8192 bytes)
0.000000] Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
0.000000] Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
0.000000] Satish: Set Processor affinity:Function 14
0.000000] Satish: Free Memory pags in all zones:Function 18
0.000000] Memory: 512MB = 512MB total
3. Debug SMP call flow

```plaintext
0.000000] Memory: 512MB = 512MB total
0.000000] Memory: 506764k/506764k available, 17524k reserved, OK highmem
0.000000] Virtual kernel memory layout:
0.000000]   vector : 0xffffffff - 0xfffff1000 (  4 kB)
0.000000]   fixmap : 0xffff0000 - 0xfffffe000 ( 896 kB)
0.000000]   vmalloc : 0xe0000000 - 0xfd000000 ( 456 MB)
0.000000]   lowmem : 0xc0000000 - 0xe0000000 ( 512 MB)
0.000000]   pkmap : 0xbfe00000 - 0xc0000000 (  2 MB)
0.000000]   modules : 0xbfe00000 - 0xbfe90000 ( 14 MB)
0.000000]   .text : 0xc0000800 - 0xc0424cb4 (4212 kB)
0.000000]   .init : 0xc0425000 - 0xc044ab40 ( 151 kB)
0.000000]   .data : 0xc044c000 - 0xc04849a0 ( 227 kB)
0.000000]   .bss : 0xc04849c4 - 0xc049b130 (  90 kB)
0.000000] Satish:Slab allocation:Function 19
0.000000] Satish:Sched_init for cpu0:Function 9
0.000000] Preemptible hierarchical RCU implementation.
0.000000] Dump stacks of tasks blocking RCU-preempt GP.
```
3. Debug SMP call flow

- 0.000000] lowmem : 0xc0000000 - 0xe0000000 (512 MB)
- 0.000000] pkmapi : 0xbfe00000 - 0xc0000000 (2 MB)
- 0.000000] modules : 0xbf000000 - 0xbfe00000 (14 MB)
- 0.000000] .text : 0xc0000000 - 0xc0424cb4 (4212 kB)
- 0.000000] .init : 0xc0425000 - 0xc044ab40 (151 kB)
- 0.000000] .data : 0xc044c000 - 0xc04849a0 (227 kB)
- 0.000000] .bss : 0xc04849c4 - 0xc049b130 (90 kB)
- 0.000000] Satish: Slab allocation: Function 19
- 0.000000] Satish: Sched_init for CPU0: Function 9
- 0.000000] Preemptible hierarchical RCU implementation.
- 0.000000] Dump stacks of tasks blocking RCU-preempt GP.
- 0.000000] RCU restricting CPUs from NR_CPUS=4 to nr_cpu_ids=2.
- 0.000000] NR_IRQS:512
- 0.000000] Satish: Init Interrupt table, GIC, traps: Function 13
- 0.000000] Zynq clock init
- 0.000000] Satish: sofirq_init: Function 15
- 0.000000] Satish: System Timer_init: Function 16
- 0.000000] xlnx, ps7-ttc-1.00.a #0 at 0xe0800000, irq=43
3. Debug SMP call flow

0.000000] Satish:Init Interrupt table,GIC,traps:Function 13
0.000000] Zynq clock init
0.000000] Satish:softirq_init:Function 15
0.000000] Satish:System_Timer_init:Function 16
0.000000] xlnx,ps7-ttc-1.00.a #0 at 0xe0000000, irq=43
0.000000] ---------[ cut here ]---------
0.000000] WARNING: at arch/arm/kernel/smp_twd.c:389 time_init+0x28/0x3c()
0.000000] twd_local_timer_of_register failed (-19)
0.000000] Modules linked in:
0.000000] [<<00131f0>] (unwind_backtrace+0x8/0xe8) from [<<001f230>] (warn_slowpath_common+0x4c/0x64)
0.000000] [<<001f230>] (warn_slowpath_common+0x4c/0x64) from [<<001f274>] (warn_slowpath_fmt+0x2c/0x3c)
0.000000] [<<001f274>] (warn_slowpath_fmt+0x2c/0x3c) from [<<0428ee0>] (time_init+0x28/0x3c)
0.000000] [<<0428ee0>] (time_init+0x28/0x3c) from [<<04256e4>] (start_kernel+0x1cc/0x31c)
0.000000] [<<04256e4>] (start_kernel+0x1cc/0x31c) from [<<00008044>] (0x8044)
0.000000] ---[ end trace 1b75b31a2719ed1c ]---
0.000000] sched_clock: 32 bits at 100 Hz, resolution 100000000ns, wraps every 4294967286ms

0.000000] Satish:Console-init:Function 17
0.000000] Console: colour dummy device 80x30
0.000000] Satish:Speed of CPU clock:Function 20
0.000000] Satish:Speed of Secondary CPU clock:Function 32

0.090000] Calibrating delay loop... 1332.01 BogoMIPS (lpj=8660096)
0.090000] pid_max: default: 32768 minimum: 301
0.090000] Mount-cache hash table entries: 512
3. Debug SMP call flow

0.000000] sched_clock: 32 bits at 100 Hz, resolution 10000000ns, wraps every 4294967296ms
0.000000] Satish:Console-init:Function 17
0.000000] Console: colour dummy device 80x30
0.000000] Satish:Speed of CPU clock:Function 20
0.000000] Satish:Speed of Secondary CPU clock:Function 32
0.090000] Calibrating delay loop... 1332.01 BogoMIPS (lpj=6660096)
0.090000] pid_max: default: 32768 minimum: 301
0.090000] Mount-cache hash table entries: 512
0.090000] Satish:RootFs Init: Function 21
0.090000] CPU: Testing write buffer coherency: ok
0.090000] Satish:Init Process creaton:Function 22
0.090000] Satish:Move data to all CPU's:Function 25
0.090000] Satish:Move & store data to each CPU:Function 33
0.090000] CPU0: thread -1, cpu 0, socket 0, mpidr 80000000
0.090000] Satish:Start Prepare CPUs:Function 24
3. Debug SMP call flow
3. Debug SMP call flow

- sched_clock: 32 bits at 100 Hz, resolution 100000000ns, wraps every 4294967286ms
- Satish: Console-init: Function 17
- Console: colour dummy device 80x30
- Satish: Speed of CPU clock: Function 20
- Satish: Speed of Secondary CPU clock: Function 32
- Calibrating delay loop... 1332.01 BogoMIPS (lpj=6660096)
- pld_max: default: 32768 minimum: 301
- Mount-cache hash table entries: 512
- Satish: RootFs Init: Function 21
- CPU: Testing write buffer coherency: ok
- Satish: Init Process creation: Function 22
- Satish: Move data to all CPU's: Function 25
- Satish: Move & store data to each CPU: Function 33
- CPU0: thread -1, cpu 0, socket 0, mpidr 80000000
- Satish: Start Prepare CPUs: Function 24
- Satish: Initialize SCU: Function 26
- hw perfevents: enabled with ARMv7 Cortex-A9 PMU driver, 7 counters available
- Satish: Allocate Initial Page tables to Each CPU1/2: Function 37
- Setting up static identity map for 0x2f1168 - 0x2f119c
- L310 cache controller enabled
- l2x0: 8 ways, CACHE_ID 0x410000c8, AUX_CTRL 0x72060000, Cache size: 524288 B
- Satish: Create New Idle Process for Each CPU1/2: Function 36
- Satish: Boot every offline CPU: Function 34
- Satish: Boot the Secondary CPU: Function 38
- Satish: Booting Secondary Processors: Function 27
- Map SCR registers
3. Debug SMP call flow

- CPU0: thread -1, cpu 0, socket 0, mpidr 80000000
- Satish: Start Prepare CPUs: Function 24
- Satish: Intilize SCU: Function 26
- hw perfevents: enabled with ARMv7 Cortex-A9 PMU driver, 7 counters available
- Satish: Allocate Intial Page tables to Each CPU1/2: Function 37
- Setting up static identity map for 0x2f1168 - 0x2f119c
- L310 cache controller enabled
- 2x0: 8 ways, CACHE_ID 0x410000c8, AUX_CTRL 0x72000000, Cache size: 524288 B
- Satish: Create New Idle Process for Each CPU1/2: Function 36
- Satish: Boot every Offline CPU: Function 34
- Satish: Boot the Secondary CPU: Function 38
- Satish: Booting Secondary Processors : Function 27
- Map SLCR registers

- Satish: Official Start of Kernel for Secondary CPUS: Function 28
- CPU1: Booted secondary processor
- Satish: CPU_Init: Function 7
- Satish: CPU_Init for Secondary CPU: Function 29
- Satish: Synchronize Boot thread and CPU0: Function 30
- Satish: Speed of CPU clock: Function 20
- Satish: Speed of Secondary CPU clock: Function 32
- Satish: Move data to all CPU's: Function 25
- Satish: Move & store data to each CPU: Function 33
- CPU1: thread -1, cpu 1, socket 0, mpidr 80000001
- Brought up 2 CPUs
3. Debug SMP call flow
3. Debug SMP call flow

```
[ 1.460000] mmcblk0: mmc0:0007 SD04G 3.70 GiB
[ 1.470000] mmcblk0: p1
[ 1.600000] adv7511-hdmi-snd adv7511_hdmi_snd.2: CODEC adv7511.2-0039 not registered
[ 1.610000] platform adv7511_hdmi_snd.2: Driver adv7511-hdmi-snd requests probe deferral
[ 1.620000] TCP: cubic registered
[ 1.620000] NET: Registered protocol family 17
[ 1.630000] VFP support v0.3: implementor 41 architecture 3 part 30 variant 9 rev 4
[ 1.630000] Registering SWP/SWPB emulation handler
[ 1.640000] registered taskstats version 1
[ 1.640000] drivers/rtc/hctsyst.c: unable to open rtc device (rtc0)
[ 1.640000] adv7511-hdmi-snd adv7511_hdmi_snd.2: CODEC adv7511.2-0039 not registered
[ 1.640000] platform adv7511_hdmi_snd.2: Driver adv7511-hdmi-snd requests probe deferral
[ 1.660000] ALSA device list:
[ 1.670000] No soundcards found.
[ 1.670000] RAMDISK: gzip image found at block 0
[ 1.980000] EXT4-fs (ram0): warning: mounting unchecked fs, running e2fsck is recommended
[ 1.990000] EXT4-fs (ram0): mounted filesystem without journal.Opts: (null)
[ 1.990000] VFS: Mounted root (ext4 filesystem) on device 1:0.
[ 2.000000] Starting init memory: 148K
[ 2.000000] Satish:Allocate Initial Page tables to Each CPU1/2:Function 37
[ 2.030000] Satish:Allocate Initial Page tables to Each CPU1/2:Function 37
```

Starting rcS...

Mounting filesystem
3. Debug SMP call flow

```bash
insmod: can't read '/lib/modules/3.6.0-digilent-13.01-00002-gaa40' 2.530000 SATISH:Allocate Initial Page tables to Each CPU
afb/pmodoled-gpio.ko: No such file or directory
[ 2.540000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
++ Exporting LEDs & SWs
[ 2.560000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
[ 2.570000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
[ 2.580000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
[ 2.580000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
[ 2.590000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
[ 2.600000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
[ 2.610000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
[ 2.610000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
[ 2.620000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
[ 2.630000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
[ 2.640000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
[ 2.640000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
[ 2.650000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
[ 2.660000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
[ 2.670000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
[ 2.670000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
rcS Complete
[ 2.680000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
zyng> ls
[ 423.750000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
[ 423.750000] SATISH:Allocate Initial Page tables to Each CPU1/2:Function 37
```
3. Debug SMP call flow

This function is keep calling and it keeps loop within this function & so this printk has been disabled in the code.
4. Usage

This session is useful in

1. Identify SMP Code flow
2. Resolve Kernel Board bring up issues
   2.1 ARM Arch. Servers having lot of clusters on the board, useful to wake up the Secondary cores.
   2.2 When Porting from one kernel version to other, debug SMP related issues

For Reference: Initial Board bring up controllers like Clocks, GPIO, UART debug flow is added using Zynq linux kernel on ZED Board in Xilinx Embedded Linux Community..
Reference: 
These documents are prepared based on Kernel Documentation of GPIO, UART, Clocks.
Thank you
Join Linaro to accelerate deployment of your Arm-based solutions through collaboration
contactus@linaro.org