LS2085 u-boot Workflow

1. Bootflow Overview of LS2

The following is LS2 Bootflow

Coming out of reset->Service processor execute bootrom->GPP bootcore executes bootrom->Bootloader at EL3 sets up DDR->Bootloader at EL2 transfers control to kernel

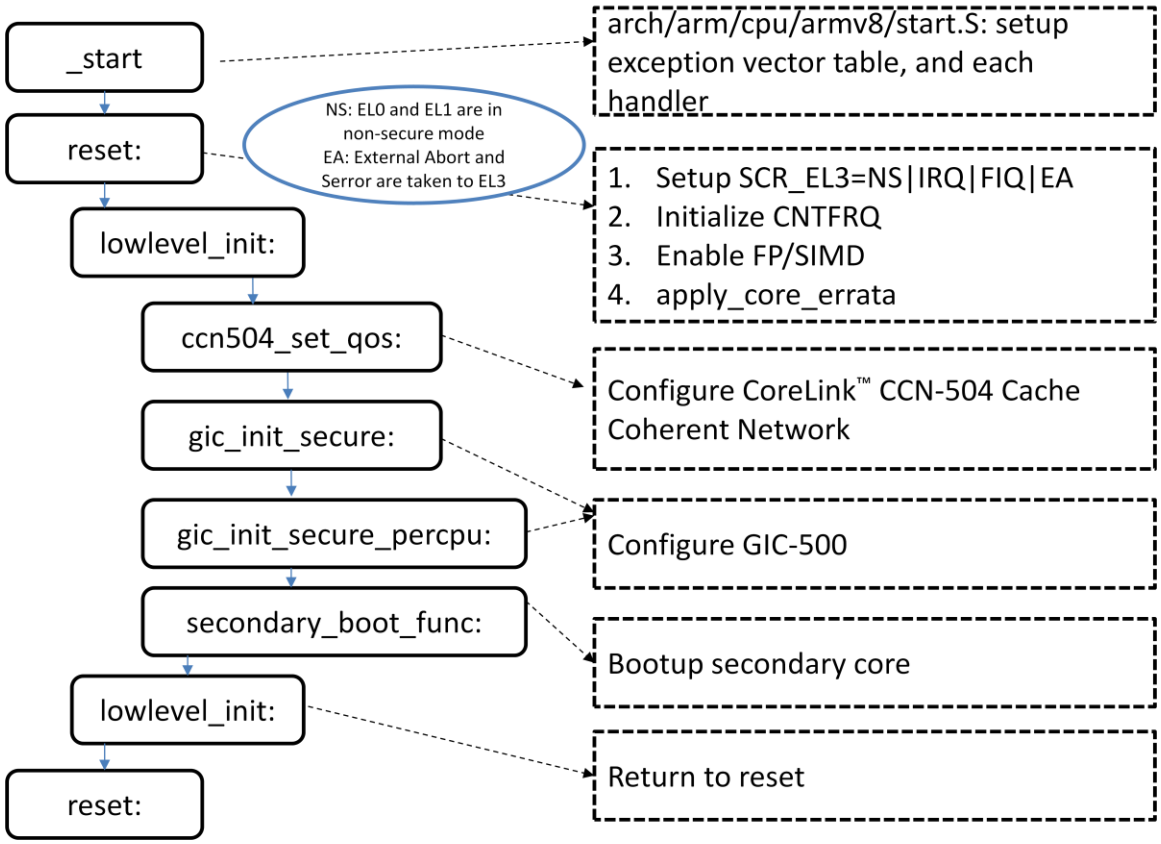
GPP Bootrom details for bootcore:

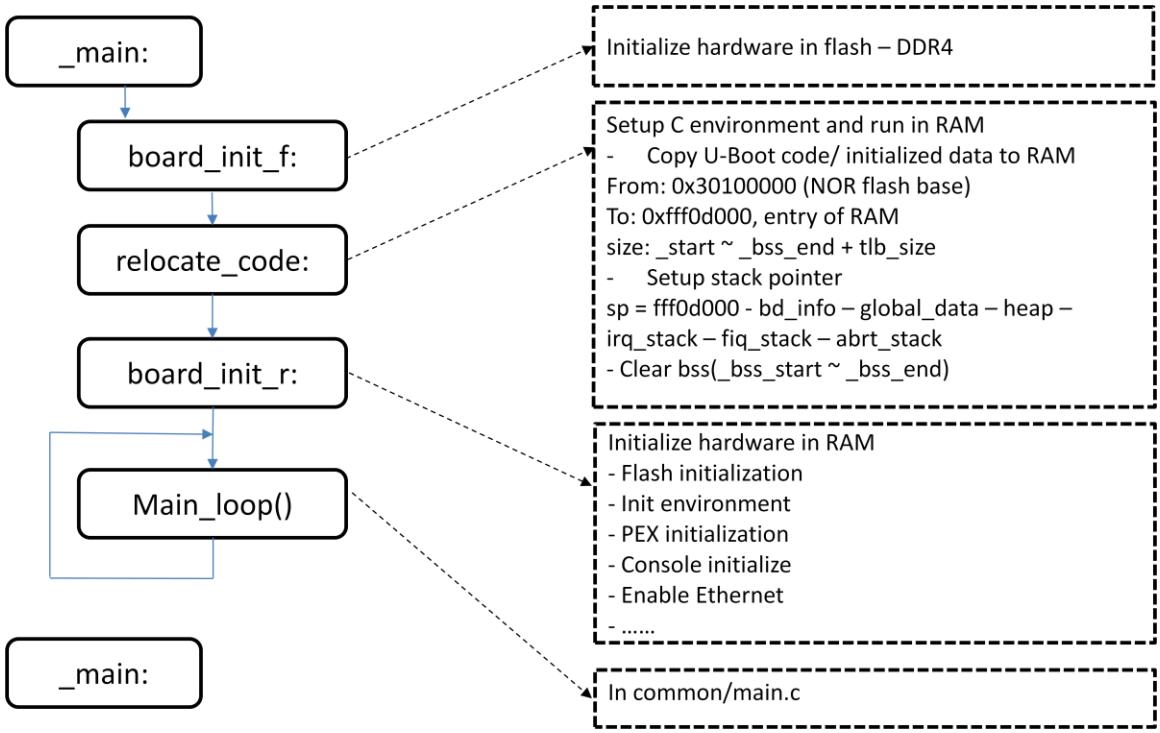
Sets L2 Cache latency->configures for personality(marks cores to be disabled)->Performs EL3 init on bootcore->Sets L3 cache in snoop mode->Gets start address of bootloader->Transers control to bootloader in EL3

GPP Bootrom details for the secondary cores:

If core to be disabled, goes through power down sequence->Sets L2 cache latency(if different cluster)->Performs EL3 init on core->Gets start address->Jumps to start address in EL3(bootloader)

2. U-BOOT Workflow for LS2085





3. LS2085QDS configuration and Initialization in U-BOOT

U-BOOT memory map for LS2085QDS

Memory map from core's view

0x00_0000_0000 .. 0x00_000F_FFFF Boot Rom

0x00_0100_0000 .. 0x00_0FFF_FFFF CCSR

0x00_1800_0000 .. 0x00_181F_FFFF OCRAM

0x00_3000_0000 .. 0x00_3FFF_FFFF IFC region #1

0x00_8000_0000 .. 0x00_FFFF_FFFF DDR region #1

0x05_1000_0000 .. 0x05_FFFF_FFFF IFC region #2

0x80_8000_0000 .. 0xFF_FFFF_FFFF DDR region #2

Other addresses are either reserved, or not used directly by u-boot.

IFC region map from core's view

During boot i.e. IFC Region #1:-

0x30000000 - 0x37ffffff : 128MB : NOR flash

0x38000000 - 0x3BFFFFFF : 64MB : Promjet

0x3C000000 - 0x40000000 : 64MB : FPGA etc

After relocate to DDR i.e. IFC Region #2:-

0x5_1000_0000..0x5_1fff_ffff Memory Hole

0x5_2000_0000..0x5_3fff_ffff IFC CSx (FPGA, NAND and others 512MB)

0x5_4000_0000..0x5_7fff_ffff ASIC or others 1GB

0x5_8000_0000..0x5_bfff_ffff IFC CS0 1GB (NOR/Promjet)

0x5_C000_0000..0x5_ffff_ffff IFC CS1 1GB (NOR/Promjet)

Processor Info Setup

arch/arm/include/asm/arch-fsl-lsch3/config.h, immap_lsch3.h

cpu/armv8/fsl-lsch3/cpu.c

#define CONFIG_SYS_INIT_SP_ADDR (CONFIG_SYS_FSL_OCRAM_BASE + 0xfff0)

#define CONFIG_SYS_IMMR 0x01000000

#define CONFIG_SYS_FSL_DDR_ADDR (CONFIG_SYS_IMMR + 0x00080000)

#define CONFIG_SYS_FSL_DDR2_ADDR (CONFIG_SYS_IMMR + 0x00090000)

#define CONFIG_SYS_FSL_DDR3_ADDR 0x08210000

#define CONFIG_SYS_FSL_GUTS_ADDR (CONFIG_SYS_IMMR + 0x00E00000)

……..

#define CONFIG_MAX_CPUS 16

#define CONFIG_SYS_FSL_IFC_BANK_COUNT 8

#define CONFIG_NUM_DDR_CONTROLLERS 3

#define CONFIG_SYS_FSL_SRDS_1

#define CONFIG_SYS_FSL_SRDS_2

MMU Setup

Files:

cpu/armv8/fsl-lsch3/cpu.c arch/arm/cpu/armv8/cache_v8.c

cpu/armv8/fsl-lsch3/cpu.c

Code Snippets:

Early MMU setup:

```
u64 *level1_table_0 = (u64 *)(CONFIG_SYS_FSL_OCRAM_BASE + 0x1000);
u64 *level1_table_1 = (u64 *)(CONFIG_SYS_FSL_OCRAM_BASE + 0x2000);
u64 *level2_table_0 = (u64 *)(CONFIG_SYS_FSL_OCRAM_BASE + 0x3000);
u64 *level2_table_1 = (u64 *)(CONFIG_SYS_FSL_OCRAM_BASE + 0x4000);
```

Final MMU setup:

```
u64 *level0_table = (u64 *)gd->arch.tlb_addr;
u64 *level1_table_0 = (u64 *)(gd->arch.tlb_addr + 0x1000);
u64 *level1_table_1 = (u64 *)(gd->arch.tlb_addr + 0x2000);
u64 *level2_table_0 = (u64 *)(gd->arch.tlb_addr + 0x3000);
u64 *level2_table_1 = (u64 *)(gd->arch.tlb_addr + 0x4000);
```

- *Setup TTBR0_Eln(base address), TCR_Eln(translation control), MAIR_Eln (attribute) (arch/arm/cpu/armv8/cache_v8.c)*

```
asm volatile("msr ttbr0_el1, %0" : : "r" (table) : "memory");
asm volatile("msr tcr_el1, %0" : : "r" (tcr) : "memory");
asm volatile("msr mair_el1, %0" : : "r" (attr) : "memory");
```

Stack pointer in OCRAM

File:

arch/arm/lib/crt0_64.S

Code Snippets:

```
ldr     x0, =(CONFIG_SYS_INIT_SP_ADDR)
sub     x0, x0, #GD_SIZE        /* allocate one GD above SP */
bic     sp, x0, #0xf    /* 16-byte alignment for ABI compliance */
mov     x18, sp                 /* GD is above SP */
```

```
    mov     x0, #0
    bl      board_init_f
```

SYSCLK

File:

board/freescale/ls2085aqds/ls2085aqds.c

Code Snippets:

```
unsigned long get_board_ddr_clk(void)
{
        u8 ddrclk_conf = QIXIS_READ(brdcfg[1]);
        switch ((ddrclk_conf & 0x30) >> 4) {
        case QIXIS_DDRCLK_100:
                return 100000000;
        case QIXIS_DDRCLK_125:
                return 125000000;
        case QIXIS_DDRCLK_133:
                return 133333333;
        }
        return 66666666;
}
```

Serial: 2 DUARTs (4 UARTs) on LS2085A, 2 serial outputs on QDS, u-boot provides driver for NS16550 compatible DUARTs.

Files:

arch/arm/include/asm/arch-fsl-lsch3/config.h

drivers/serial/ns16550.c, serial.c

Code Snippets:

```
/* Serial Port */
```

#define CONFIG_CONS_INDEX 1

#define CONFIG_SYS_NS16550

#define CONFIG_SYS_NS16550_SERIAL

#define CONFIG_SYS_NS16550_REG_SIZE 1

#define CONFIG_SYS_NS16550_CLK (get_bus_freq(0)/2)

#define CONFIG_BAUDRATE 115200

#define CONFIG_SYS_BAUDRATE_TABLE { 9600, 19200, 38400, 57600, 115200 }

#define CONFIG_SYS_NS16550_COM1 (CONFIG_SYS_IMMR + 0x011C0500)

#define CONFIG_SYS_NS16550_COM2 (CONFIG_SYS_IMMR + 0x011C0600)

DDR Setup: Supports multiple controllers and SPD-based (I2C eeprom) initialization Supports interleaving.

Files:

drivers/ddr/fsl/ddr4_dimm_params.c, fsl_ddr_gen4.c

board/freescale/ls2085aqds/ddr.c ddr.h(board specific parameters)

Code Snippets:

#define CONFIG_SYS_DDR_RAW_TIMING

#define CONFIG_DDR_SPD

#define CONFIG_DDR_ECC

#define CONFIG_ECC_INIT_VIA_DDRCONTROLLER

#define SPD_EEPROM_ADDRESS1 0x51

#define SPD_EEPROM_ADDRESS2 0x52

static const struct board_specific_parameters udimm0[] = {

/

** memory controller 0*

** num| hi| rank| clk| wrlvl | wrlvl | wrlvl*

** ranks| mhz| GB |adjst| start | ctl2 | ctl3*

\/

```
        {2,  1350, 0, 4,     6, 0x0708090B, 0x0C0D0E09,},
        {2,  1666, 0, 4,     7, 0x08090A0C, 0x0D0F100B,},
        {2,  1900, 0, 4,     7, 0x09090B0D, 0x0E10120B,},
        {2,  2200, 0, 4,     8, 0x090A0C0F, 0x1012130C,},
        {}
};
```

Relocate to DDR Space: Relocates u-boot code to top of memory U-boot is Relocate to the last 1MB,Set up of stack space, continues to run from DDR after relocation.

Files:arch/arm/lib/relocate_64.S

Code Snippets:

```
ENTRY(relocate_code)
        stp     x29, x30, [sp, #-32]!   /* create a stack frame */
        mov     x29, sp
        str     x0, [sp, #16]
        /*
         * Copy u-boot from flash to RAM
         */
        ldr     x1, =__image_copy_start /* x1 <- SRC &__image_copy_start */
        subs    x9, x0, x1              /* x9 <- relocation offset */
        b.eq    relocate_done           /* skip relocation */
        ldr     x2, =__image_copy_end   /* x2 <- SRC &__image_copy_end */
```