Porting dual core STM32H7 in Zephyr RTOS

Erwan Gouriou
Assignee in LiTE group
STMicroelectronics
A bit of context

- stm32h747 dual core basic support in zephyr was upstreamed mid ‘19
- Initial focus was made on enabling boot on each core with basic applications
- This work showed challenges in current Zephyr state to allow fully safe and functional usage on multi core chips

- This presentation does not pretend to be exhaustive on STM32H747 Dual core architecture, aim is to provide basis for development on Zephyr and understanding on the next required steps.
Agenda

● stm32h747 introduction, essential parts to play with both cores
  ○ RCC
  ○ HSEM
  ○ Extended Interrupt controller
  ○ Option bytes
● Board porting: Delta with single core typical board
  ○ dts, Kconfig
  ○ Take away
● Resource sharing at run time
● Boot options
● Plans for Inter-core communication
Quick target introduction

● The (dual) core: STM32H747
  ○ Cortex-M7 and Cortex-M4, each one with dedicated NVIC
  ○ Up to 480MHz on C-M7, 240MHz on C-M4
  ○ Reset and Clock Controller capable of peripheral allocation
  ○ Hardware Semaphore

● The board: STM32H747I-DISCO:
  ○ 2M Flash, 1M of RAM
  ○ 2x512 Quad-SPI NOR Flash
  ○ 256M SDRAM
  ○ 4” capacitive touch LCD display module with MIPI DSI
  ○ Ethernet, USB OTG HS
  ○ MEMS digital microphone, audio codec
  ○ Full range of connectors (Arduino V3, Camera, SPDIF, Jack, ...)
  ○ ST-Link V3
STM32H747 Reset and Clock Control

- Capable of peripheral allocation through CPU dedicated registers:
  - RCC_C1_PERxEN: PERx clock gating for CPU1
  - RCC_C2_PERxEN: PERx clock gating for CPU2

- Common registers accessible by both cores (RCC_CR)
- Core dedicated registers mapped at the same address (RCC_PERxEN):
  - When accessed by CPU1, CPU1 register bank is accessed
  - When accessed by CPU2, CPU2 register bank is accessed

- CPU1 register bank mapped at a CPU1 accessible address (RCC_C1_PERxEN)
- CPU2 register bank mapped at a CPU2 accessible address (RCC_C2_PERxEN)
STM32H747 Dual Core clock tree
STM32H747 Hardware semaphore (HSEM)

- 32 (32-bit) register based semaphores.
- Ensure synchronization between different processes (across or within cores)
- Provides a non blocking mechanism to lock semaphores in an atomic way
- Interrupt generation when a semaphore is freed (2 interrupts lines)
- Semaphore clear protection
- Global semaphore clear per COREID
STM32H747 Extended Interrupt Controller

- IP provides interrupts and event configuration (mostly used for GPIO interrupt control for now)
- Core agnostic registers allow IRQ configuration (raising/falling edge)
- Core dedicated registers allow IRQ/Event masking per core
  - EXTI_C1IMR1/IMR1: IRQ mask cfg for CPU1
  - EXTI_C2IMR1: IRQ mask cfg for CPU2

- **All registers are accessible from both Cores**

<table>
<thead>
<tr>
<th>Base Address</th>
<th>Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTI_C2_BASE</td>
<td>C2_EMR1</td>
</tr>
<tr>
<td></td>
<td>C2_IMR1</td>
</tr>
<tr>
<td>EXTI_C1_BASE</td>
<td>EMR1/C1_EMR1</td>
</tr>
<tr>
<td></td>
<td>IMR1/C1_IMR1</td>
</tr>
<tr>
<td>EXTI_BASE</td>
<td>FTSR</td>
</tr>
<tr>
<td></td>
<td>RTSR</td>
</tr>
</tbody>
</table>
STM32H747 Option bytes

- On STM32 SoCs, option bytes are user settable options, located in Flash
  - Persistent across switch off/on cycles
  - Require specific write procedure to be set/reset
  - Could be read/updated using STM32CubeProgrammer tool

- Dual core specific option bytes
  - BOOT_CM7: Controls C-M7 boot at switch on (don’t touch this one!!)
  - BOOT_CM4: Controls C-M4 boot at switch on
  - BOOT_CM7_ADD0/1: Default 0x8000/0x1FF0
  - BOOT_CM4_ADD0/1: Default 0x8100/0x1000
STM32H7 Board configuration: Dual core impact

- Configuring a dual core target brings in some new constraints
- They need to be dealt with minimum code additions for:
  - Maintenance purpose
  - Compatibility with existing single core configurations or targets

- We’ll see in next slides how this is currently implemented
Typical single core Zephyr board/

- boards/arm/my_board
  - doc/
  - support/
  - board.cmake
  - CMakeLists.txt
  - Kconfig.board
  - Kconfig.defconfig
  - pinmux.c
  - board_defconfig
  - board.dts
  - board.yaml
stm32h747i_disco

- boards/arm/stm32h747i_disco
  - doc/
  - support/
  - board.cmake
  - CMakeLists.txt
  - Kconfig.board
  - Kconfig.defconfig
  - pinmux.c
  - board_defconfig
  - board.defconfig
  - board.dts
  - board.yaml

This part is untouched for dual core target
Typical single core Zephyr board dts

- **boards/arm/my_board**
  - `doc/`
  - `support/`
  - `board.cmake`
  - `CMakeLists.txt`
  - `Kconfig.board`
  - `Kconfig.defconfig`
  - `pinmux.c`
  - `board_defconfig`
  - `board.dts`
  - `board.yaml`

- **stm32XX.dtsi**
  - Defines common base of STM32XX series resources

- **stm32XXyy.dtsi**
  - Defines XXyy additional soc resources

- **stm32XXyyXz.dtsi**
  - Defines Flash size

- **board.dts**
  - Includes stm32XXyyXz.dtsi
  - Describes and enables BSP resources
  - Enables assigned resources
stm32h747i_disco dts

- boards/arm/stm32h747i_disco
  - doc/
  - support/
  - board.cmake
  - CMakeLists.txt
  - Kconfig.board
  - Kconfig.defconfig
  - pinmux.c
  - board_defconfig
  - board.dts
  - board.yaml

- stm32h7.dtsi
  - Defines common base of STM32H7 series resources
  - Defines both m4 and m7 cpu nodes

- stm32h747.dtsi
  - Defines soc resources

- stm32h747Xi_m7.dtsi
  - Deletes m4 cpu node
  - Assigns M7 Flash size and based address

- stm32h747i_disco.dtsi
  - Describes common bsp resources, with status = “disabled”
  - Leds, buttons, ..

- stm32h747i_disco_m7.dtsi
  - Includes stm32h747Xi_m7.dtsi
  - Includes stm32h747i_disco.dtsi
  - Enables assigned resources
Typical Zephyr board Kconfig’ery (1 / 3)

- boards/arm/my_board
  - doc/
  - support/
  - board.cmake
  - CMakeLists.txt
  - Kconfig.board
  - Kconfig.defconfig
  - pinmux.c
  - board_defconfig
  - board.dts
  - board.yaml

Kconfig board symbol definition:

```
config BOARD_MY_BOARD
    bool "My Development Board"
    depends on SOC_MYSOC
```
Typical Zephyr board Kconfig’ery (2 / 3)

- `boards/arm/my_board`
  - `doc/`
  - `support/`
  - `board.cmake`
  - `CMakeLists.txt`
  - `Kconfig.board`
  - `Kconfig.defconfig`
  - `pinmux.c`
  - `board_defconfig`
  - `board.dts`
  - `board.yaml`

  Used for board **conditional configuration**: essential for HW abstraction at application level

  ```
  if SPI
  config SPI_1
      default y
  endif
  ```
Typical Zephyr board Kconfig’ery (3 / 3)

- boards/arm/my_board
  - doc/
  - support/
  - board.cmake
  - CMakeLists.txt
  - Kconfig.board
  - Kconfig.defconfig
  - pinmux.c
  - board_defconfig
  - board.dts
  - board.yaml

  Used for board **typical configuration**. Application independent boards settings, typically:

  ```
  # Target general characteristics
  CONFIG_ARM=y, CONFIG_BOARD_NUCLEO_F401RE=y
  CONFIG_SOC_SERIES_STM32F4X=y,
  CONFIG_SOC_STM32F401XE=y
  CONFIG_CORTEX_M_SYSTICK=y, CONFIG_ARM_MPU=y

  # Basic drivers activation
  CONFIG_PINMUX=y, CONFIG_SERIAL=y, CONFIG_GPIO=y,
  CONFIG_CLOCK_CONTROL=y, CONFIG_UART_CONSOLE=y,
  CONFIG_CONSOLE=y

  # Clock configuration
  CONFIG_SYS_CLOCK_HW_CYCLES_PER_SEC=84000000,
  CONFIG_CLOCK_STM32_HSE_CLOCK=80000000,
  CONFIG_CLOCK_STM32_SYSCLK_SRC_PLL=y ...
  ```
stm32h747i_disco Kconfig’ery (1 / 5)

- boards/arm/stm32h747i_disco
  - doc/
  - support/
  - board.cmake
  - CMakeLists.txt
  - Kconfig.board
    - Kconfig.defconfig
    - pinmux.c
    - board_defconfig
    - board.dts
    - board.yaml

```c
config BOARD_STM32H747I_DISCO_M7
  bool "STM32H747I Discovery Board"
  depends on SOC_STM32H747XX
  select CPU_CORTEX_M7
```

```c
config BOARD_STM32H747I_DISCO_M4
  bool "STM32H747I Discovery Board"
  depends on SOC_STM32H747XX
  select CPU_CORTEX_M4
```
stm32h747i_disco Kconfig’ery (2 / 5)

- boards/arm/stm32h747i_disco
  - doc/
  - support/
  - board.cmake
  - CMakeLists.txt
  - Kconfig.board
  - Kconfig.defconfig
  - pinmux.c
  - board_defconfig
  - board.dts
  - board.yaml

Used for 2 purposes:
- common resources configuration
- static resources allocation (C-M4/C-M7)

# Single/dual core configurations
config STM32H7_DUAL_CORE (def_bool y/n)

# Dual core boot configuration
if STM32H7_DUAL_CORE
  choice STM32H7_DUAL_CORE_BOOT
    # Use out of the box config by default
    # default STM32H7_BOOT_CM4_CM7
    default STM32H7_BOOT_CM7_CM4GATED
  endchoice
endif # STM32H7_DUAL_CORE
stm32h747i_disco Kconfig’ery (3 / 5)

- boards/arm/stm32h747i_disco
  - doc/
  - support/
  - board.cmake
  - CMakeLists.txt
  - Kconfig.board
  - Kconfig.defconfig
  - pinmux.c
  - board_defconfig
  - board.dts
  - board.yaml

Used for 2 purposes:
- common resources configuration
- static resources allocation (C-M4/C-M7)

```
# Clock configuration
config SYS_CLOCK_HW_CYCLES_PER_SEC
default 400000000 if BOARD_STM32H747I_DISCO_M7
default 200000000 if BOARD_STM32H747I_DISCO_M4

config CLOCK_STM32_HPRE
  # HCLK: 200MHz
  default 2
```
stm32h747i_disco Kconfig’ery (4 / 5)

- boards/arm/stm32h747i_disco
  - doc/
  - support/
  - board.cmake
  - CMakeLists.txt
  - Kconfig.board
  - Kconfig.defconfig
  - pinmux.c
  - board_defconfig
  - board.dts
  - board.yaml
  
  Used for 2 purposes:
  - common resources configuration
  - static resources allocation (C-M4/C-M7)

  if SERIAL
  
  config UART_8
default y if BOARD_STM32H747I_DISCO_M7
  
  endif # SERIAL
stm32h747i_disco Kconfig’ery (5 / 5)

- boards/arm/stm32h747i_disco
  - doc/
  - support/
  - board.cmake
  - CMakeLists.txt
  - Kconfig.board
  - Kconfig.defconfig
  - pinmux.c
  - board_defconfig_m4
  - board_defconfig_m7
  - board.dts
  - board.yaml

  One file per target (core). **Usage is minimized!!**

  # target general characteristics
  CONFIG_ARM=y, CONFIG_BOARD_STM32H747I_DISCO_M4=y
  CONFIG_SOC_SERIES_STM32H7X=y, CONFIG_soc_STM32H747XX=y
  CONFIG_CORTEX_M_SYSTICK=y

  # Basic drivers activation:
  CONFIG_PINMUX=y, CONFIG_GPIO=y, CONFIG_CLOCK_CONTROL=y
Board configuration take away

Static resources allocation needed for Dual core configuration request a different usage of board configuration:

- Resources assignment by Kconfig could be done but is heavy and will likely be removed with better device tree integration in Zephyr
- Could be done by dts but, to check coherent resources allocation between cores in dts, a tool would be needed

This highlights the need for a System device tree tool
Run time resource sharing

Some resources need to be accessed by both core at run time:

- **clock-control**
  - Configuration is done by C-M7 before C-M4 needs it
  - Configuration code is not accessible by C-M4
  - Clock activation/deactivation: Accessed through common mapped registers with core abstraction API

- **GPIO**
  - Need to be accessed from both cores in run time
  - No specific HW protection available
  - Only configuration is protected using HSEM

- **Extended Interrupt controller**
  - Each core should access its own registers
  - No core abstraction API available right now. Only CPU1 access available
Boot options: 3 options

● Single Core Boot
  ○ Usual single core boot

● Gated boot
  ○ C-M7 triggers C-M4 boot when ready

● Concurrent boot
  ○ C-M7 and C-M4 boot both at power up
  ○ C-M4 enters STOP mode after init and then WFE
  ○ C-M7 generates HSEM IRQ when ready

● CM-7 boots as usual
  ○ Kernel init
  ○ Clock control configuration
  ○ Drivers init

● Before application starts:
  ○ Gated boot: Force CM-4 boot
  ○ Concurrent boot: Generate HSEM IRQ
  ○ Single boot: Nothing
Inter-core communication on STM32H7/Zephyr

- Targeted framework: OpenAMP
- In Zephyr, current implementation requests a IPM (Inter processor Mailbox) subsystem
- Current IPM STM32 driver, used on STM32MP1
  - Based on dedicated IPCC IP
  - Not available on STM32H7
- STM32H7 IPM implementation should rely on hardware semaphore IP (HSEM)
  - No zephyr driver/API for this today
  - HSEM should also be used for real hwspinlock purposes
- Need a solution to address both needs. 2 options:
  - HSEM based HWSpinlock and HSEM based IPM
  - HSEM based HWSpinlock and HWSpinlock based IPM
Wrap up

Things that we currently miss for great AMP Dual Core experience with STM32H7 on Zephyr:

- Reliable way to split static resources between cores (system device tree)
- Complete the resource definition transition from Kconfig to dts
- Zephyr HWspinlock API
- Find a way to use HSEM both for Zephyr IPM(Mailbox) and HWSpinlock
Questions ?
Thank you

Join Linaro to accelerate deployment of your Arm-based solutions through collaboration

contactus@linaro.org