pse-blinky

This example project demonstrates how to use GPIO and UART in the PolarFire SoC Renode emulation.

Only harts 0 and 1 are used. The harts2-4 are forcefully halted from within Renode by the launcher with these commands:

```
monitor sysbus.u54_2 IsHalted true
monitor sysbus.u54_3 IsHalted true
monitor sysbus.u54_4 IsHalted true
```

Project's preprocessor define **MPFS_HAL_LAST_HART=1** makes sure that HAL is aware that the harts2-4 are not present. Targeting real HW with all 5 harts enabled will require to change this define (Duplicating the Debug configuration with separate HW related tweaks will give most flexibility). If only one configuration is desired then it's possible to remove the **monitor sysbus.u54_2 IsHalted true** commands and remove the **MPFS_HAL_LAST_HART=1**. This aproach will work on both Renode and a real HW, but the Renode performance might suffer as it is prefered to keep as many cores and as frequently as possible inside sleep/halted state (emulating infinite loops will decrease the emulation performance).

To debug/run the demo launch the **pse-blinky Renode hart0 Start-platform-and-debug** launcher.

All harts have their own separate main function:

- **e51(void)** in /src/application/hart0/e51.c
- **void u54_1(void)** in /src/application/hart1/u54_1.c
- **void u54_2(void)**, **void u54_3(void)** and **void u54_4(void)** are not implemented and would typically fallback to HAL's *weak* implementation and keep the harts inside infinite loops. However, from within Renode these harts are forcefully halted and therefore will not even reach the infinite loops.

Use Firmware Catalog to get examples with all harts defined

Why group launchers are used

Group launchers are used to simplify the Renode launching sequence. It's simpler to use a group launcher, but if needed all the steps can be executed manually: - Launching the Renode emulation framework with the correct platform selected (therefore there are different launchers for Mi-V and PF SoC) - Waiting for it to

initialize, setup GDB servers and be ready for the GDB client - Then trigger the correct GDB debug launcher which will connect with the GDB client to the GDB server running inside the Renode.

Executing example in the Renode emulation

Each launcher can attach to only one hart. Each hart has own GDB port:

- hart 0 e51 at port 3333
- hart 1 u54_1 at port 3334
- hart 2 u54_2 at port 3335
- hart 3 u54_3 at port 3336
- hart 4 u54_4 at port 3337

Build the project and run **Launch Group** named **pse-blinky Renode hart0 Start-platform-and-debug** to start the emulation and attach to hart0. It is possible to duplicate the **pse-blinky Renode hart0 Debug** launcher and connect to different GDB ports (harts). The example launcher is halting harts 3 and 4 as they are not used in the example. If all harts are needed then remove the `monitor u54_4 IsHalted true` commands from the launcher and adjust the **MPFS_HAL_LAST_HART** preprocessor define.

When loading the ELF file to a specific hart the PCs for other harts will not be set correctly and therefore the first launcher needs to call these monitor commands. The `pse-blinky Renode hart0 Debug` example sets all the u54 hart's PCs to the PC of the e51 hart:

```
monitor $COMMON_PC=‘sysbus.e51 PC‘
monitor runMacro $SetAllPCs
monitor start
```

Troubleshooting

When troubleshooting applications which interact with the peripherals, then increased verbosity can improve the visibility into the peripheral. Increasing log level in the debug launcher can show events happening in the peripheral. For increasing PLIC and GPIO log verbosity the following commands has to be added to the Launcher -> Startup -> Initialization commands:

```
monitor logLevel -1 sysbus.gpio1
monitor logLevel -1 sysbus.plic
```

The log levels are:

- NOISY (-1)
- DEBUG (0)
- INFO (1)
- WARNING (2)
- ERROR (3)

For more information and useful commands read the Renode's documentation. These commands can be typed in the Renode's monitor windows or can be sent from the SoftConsole's debug launcher through the GDB with a **monitor** prefix.

Connecting to other harts

Bundled example launcher `pse-blinky Renode hart1 Attach-to-running` will connect to the platform set the PCs in a similar manner as the hart0 launcher would do (in this case the PC is read from hart1) and is using different gdb port to select the desired hart. This is useful when doing single hart debugging. While being able to debug a specific hart. The other harts are executing code as they normally would, just the breakpoints on their code will not work. Breaking/suspending connected hart will suspend the execution of the other harts as well.

Connecting to multiple harts at the same time

This feature is not fully supported yet, but sometimes a workaround can be used. Use it at own risk, the workaround is not a replacement for stable multicore debug functionality.

Start Renode and connect to the hart0 with the **pse-blinky Renode hart0 Start-platform-and-debug** group launcher. Let the hart0 running (do not try this with halted hart0) and configure a new launcher as follows:

- Debugger tab:
  - Leave the **Start OpenOCD locally** unchecked
  - Select a new port to connect to (3333 to 3337 for hart0 - 4)
- Startup tab:
  - Uncheck **Initial Reset**

- Check **Load symbols**
- Uncheck **Load executable**
- Uncheck **Set breakpoint at**
- Check **Continue**
- Remove uneeded **Intial Reset** and **Run/Restart commands** monitor commands. Do **NOT** set the PCs as they were set when the first launcher connected
- Add new commands as required, for example showing new terminal `monitor showAnalyzer sysbus.mmuart1`

Run the second launcher connecting to the same Renode platform instance.

For demonstration purposes two launchers were configured in this way. The **pse-blinky Renode hart1 Attach-to-running** will connect to the platform, but expects previous launcher (pse-blinky Renode hart0 Start-platform-and-debug / pse-blinky Renode hart0 Debug) already having the PCs set, loaded the executable ELFfile and left the platform **running** (do **NOT** use it with a halted/suspended platform). In contrast to a typical debug launcher, this type of launcher will not update the editor windows as the launcher "continued" the execution. After connecting the **Suspend** button has to be clicked to halt/suspend the emulation to see where the hart1 is currently. It might be in some nested function (or loop) and probably user will have to add new breakpoint just to get into an upper level of code, or to the desired location. Setting u54_1 as the main breakpoint for these launchers might not work as the harts started at the same time and the main breakpoint is very likely already passed even before the launcher tries to connect to the hart. Trying to break on unreachable breakpoints will make the GDB client indefinitely to wait for it.

Extra attention needs to be made to have the correct GDB client selected (Window -> Show view -> Debug, typically located in the bottom left) as the commands (continue/halt/step over etc..) are sent to the selected GDB client (specific hart).

Work in progress

The Renode is work in progress and some SoftConsole's commands (such as RTOS queries) are not supported yet. This can result in an increased number of warning or error messages (many of which are safe to ignore).

More information

For more details read Renode documentation, SoftConsole Release Notes and visit: https://www.microsemi.com/product-directory/fpga-soc/5210-mi-v-embedded-ecosystem#renode-webinar-series