Laboratory 3 report

Cédric Hölzl Antoine Brunner

May 2020

1 Introduction

The goal of this laboratory was to build a complete Linux system on a DE0-nano-SoC - a hybrid board consisting of a FPGA and an ARM dual-core HPS. Building a Linux system tailored to a specific hardware architecture is not an easy task, and there were a lot of steps involved before succeeding with that. First, we had to use the QSys platform designer to configure the hardware properly. Then, we had to use Altera's BSP editor to generate a boot loader tailored to the hardware configuration. After that, we had to compile the Linux OS and flash it to an SD card, along with the pre-loader and the boot loader. Finally, we could boot the system, and log onto it using a minicom terminal and after configuring the wired interface via SSH from a remote device.

2 Configuring the hardware using QSys

In this part, we had to configure the hardware for the Linux system. While the Linux system can run on an ARM processor, we still had to configure the FPGA part, because the goal was to reuse some hardware interfaces from previous labs. That is, we want to configure the joysticks and the thermal camera so that they can be used by the HPS. Figure 1 shows how we configured the system with the joysticks, the thermal camera and the ARM processor.

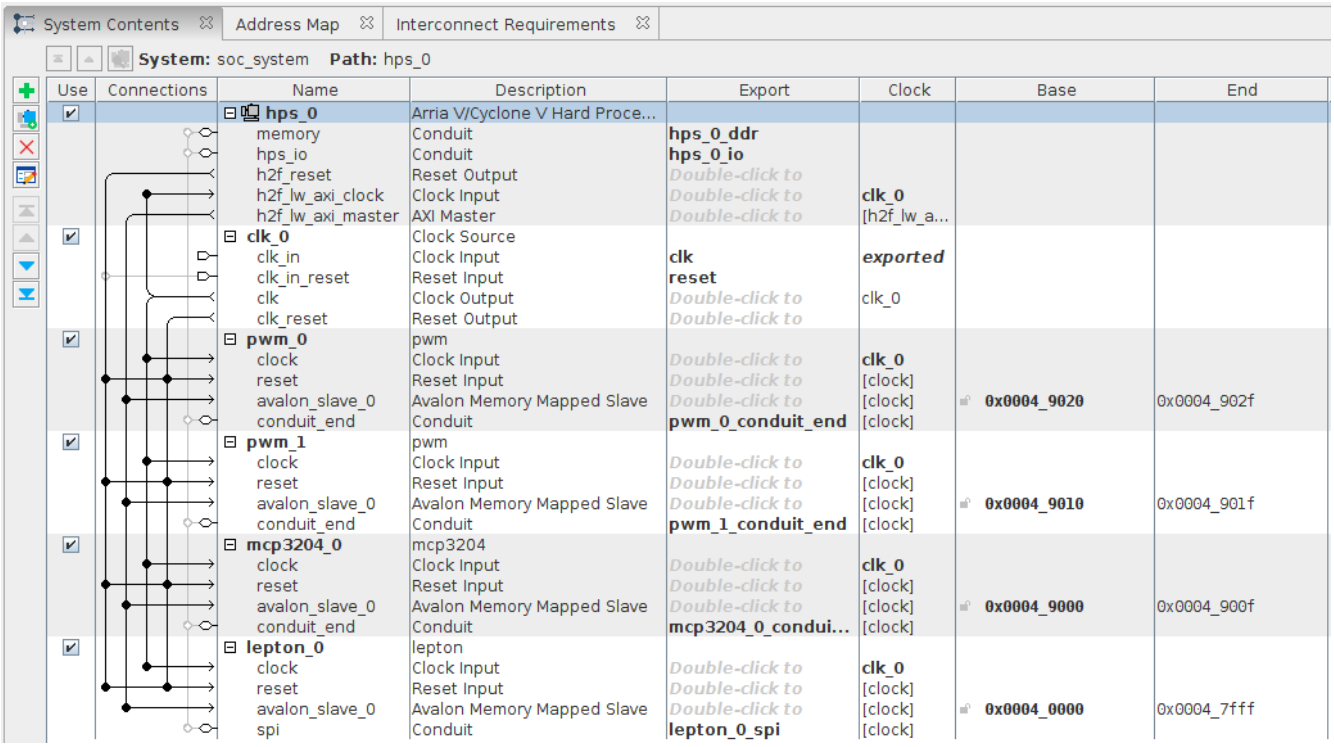

Figure 1: Overall view of the system

As a side note, since the HPS would run a Linux OS, accessing the memory mapped peripherals cannot be done in the virtual address space of the processes. Several solutions exists, such as creating a driver, or probably simpler, using *mmap* to map physical pages into the virtual address space. That was not the concern of the lab, but it is worth mentioning it, because that's a problem we will face in the next laboratory.

An error that we made while using QSys was that we didn't understand what the *peripheral multiplexer* tab was doing. Due to that misunderstanding, we did not select the pins that system needed, and couldn't boot the system correctly. After an explanation by an assistant, we finally understood how that pin multiplexer works, and we could fix the mistake that we had made.

Other than that, we also had to instantiate the system in the top-level VHD file, by making the right pin assignments for all the components, and get the design to compile. We also had some problems at this stage. Initialy we left some pins unassigned by mistake, causing the fitter to fail, but that was easily fixed.

3 Compiling the system

To compile the system on the board, we still had a long way. We had to generate the pre-loader using the BSP editor. After that, we downloaded Alteras's U-boot, configured it for the ARM processor and compiled it. The last piece of the puzzle was to download Ubuntu 14, compile it, and configure it by creating scripts that would run at the first start.

Although we have not succeeded initialy in doing those steps by ourselves, we didn't encounter any major problems in those steps, thanks to the Soc-FPGA Design Guide.

The final step was to put all the generated files in the right partitions of the SD card, so that the system could properly boot: the pre-loader to the A2 partition, the boot loader to the FAT32

partition and the Linux file system to the EXT3 partition.

We were already familiar with creating partitions and moving data to certain partitions using the command line, so we managed to do this step without major difficulties.

4 Problems encountered

Of course, it would have been to easy if everything went correctly... We didn't manage to boot our system from the first time because of a very dumb mistake. At least, we have learnt something from that mistake.

The mistake that we did is that we forgot to enable the GPIO pins in the QSys platform designer, meaning that the multiple ports (SD, Ethernet, ...) would never be properly connected to the processor. Figure 2 shows the interface on which we should have selected the right pins. That was the reason why we could not see anything in the Minicom terminal when the system was supposed to be booting. That mistake came from the fact that we didn't understand the QSys interface very well, nor the utility of the pin multiplexer. After an explanation from one of the assistants we have now understood what this interface is useful for, and we managed to fix the problem pretty easily.

Peripherals Mux Table

RGMII0_TX_CLK			EMAC0.TX_CLK (Set0)	GPIO00	LOANIO00
RGMII0_TXD0		**USB1.D0 (Set0)**	EMAC0.TXD0 (Set0)	GPIO01	LOANIO01
RGMII0_TXD1		**USB1.D1 (Set0)**	EMAC0.TXD1 (Set0)	GPIO02	LOANIO02
RGMII0_TXD2		**USB1.D2 (Set0)**	EMAC0.TXD2 (Set0)	GPIO03	LOANIO03
RGMII0_TXD3		**USB1.D3 (Set0)**	EMAC0.TXD3 (Set0)	GPIO04	LOANIO04
RGMII0_RXD0		**USB1.D4 (Set0)**	EMAC0.RXD0 (Set0)	GPIO05	LOANIO05
RGMII0_MDIO	I2C2.SDA (Set0)	**USB1.D5 (Set0)**	EMAC0.MDIO (Set0)	GPIO06	LOANIO06
RGMII0_MDC	I2C2.SCL (Set0)	**USB1.D6 (Set0)**	EMAC0.MDC (Set0)	GPIO07	LOANIO07
RGMII0_RX_CTL		**USB1.D7 (Set0)**	EMAC0.RX_CTL (Set0)	GPIO08	LOANIO08
RGMII0_TX_CTL			EMAC0.TX_CTL (Set0)	**GPIO09**	LOANIO09
RGMII0_RX_CLK		**USB1.CLK (Set0)**	EMAC0.RX_CLK (Set0)	GPIO10	LOANIO10
RGMII0_RXD1		**USB1.STP (Set0)**	EMAC0.RXD1 (Set0)	GPIO11	LOANIO11
RGMII0_RXD2		**USB1.DIR (Set0)**	EMAC0.RXD2 (Set0)	GPIO12	LOANIO12
RGMII0_RXD3		**USB1.NXT (Set0)**	EMAC0.RXD3 (Set0)	GPIO13	LOANIO13
NAND_ALE	QSPI.SS3 (Set1) (Set0)	**EMAC1.TX_CLK (Set0)**	NAND.ALE (Set0)	GPIO14	LOANIO14
NAND_CE	USB1.D0 (Set1)	**EMAC1.TXD0 (Set0)**	NAND.CE (Set0)	GPIO15	LOANIO15
NAND_CLE	USB1.D1 (Set1)	**EMAC1.TXD1 (Set0)**	NAND.CLE (Set0)	GPIO16	LOANIO16
NAND_RE	USB1.D2 (Set1)	**EMAC1.TXD2 (Set0)**	NAND.RE (Set0)	GPIO17	LOANIO17
NAND_RB	USB1.D3 (Set1)	**EMAC1.TXD3 (Set0)**	NAND.RB (Set0)	GPIO18	LOANIO18
NAND_DQ0		**EMAC1.RXD0 (Set0)**	NAND.DQ0 (Set0)	GPIO19	LOANIO19
NAND_DQ1	I2C3.SDA (Set0)	**EMAC1.MDIO (Set0)**	NAND.DQ1 (Set0)	GPIO20	LOANIO20
NAND_DQ2	I2C3.SCL (Set0)	**EMAC1.MDC (Set0)**	NAND.DQ2 (Set0)	GPIO21	LOANIO21
NAND_DQ3	USB1.D4 (Set1)	**EMAC1.RX_CTL (Set0)**	NAND.DQ3 (Set0)	GPIO22	LOANIO22
NAND_DQ4	USB1.D5 (Set1)	**EMAC1.TX_CTL (Set0)**	NAND.DQ4 (Set0)	GPIO23	LOANIO23
NAND_DQ5	USB1.D6 (Set1)	**EMAC1.RX_CLK (Set0)**	NAND.DQ5 (Set0)	GPIO24	LOANIO24
NAND_DQ6	USB1.D7 (Set1)	**EMAC1.RXD1 (Set0)**	NAND.DQ6 (Set0)	GPIO25	LOANIO25
NAND_DQ7		**EMAC1.RXD2 (Set0)**	NAND.DQ7 (Set0)	GPIO26	LOANIO26
NAND_WP	QSPI.SS2 (Set1) (Set0)	**EMAC1.RXD3 (Set0)**	NAND.WP (Set0)	GPIO27	LOANIO27
NAND_WE		QSPI.SS1 (Set0)	NAND.WE (Set0)	GPIO28	LOANIO28
QSPI_IO0	USB1.CLK (Set1)		QSPI.IO0 (Set1) (Set0)	GPIO29	LOANIO29
QSPI_IO1	USB1.STP (Set1)		QSPI.IO1 (Set1) (Set0)	GPIO30	LOANIO30
QSPI_IO2	USB1.DIR (Set1)		QSPI.IO2 (Set1) (Set0)	GPIO31	LOANIO31
QSPI_IO3	USB1.NXT (Set1)		QSPI.IO3 (Set1) (Set0)	GPIO32	LOANIO32
QSPI_SS0			QSPI.SS0 (Set1) (Set0)	GPIO33	LOANIO33
QSPI_CLK			QSPI.CLK (Set1) (Set0)	GPIO34	LOANIO34
QSPI_SS1			QSPI.SS1 (Set1)	**GPIO35**	LOANIO35
SDMMC_CMD		USB0.D0 (Set0)	**SDIO.CMD (Set0)**	GPIO36	LOANIO36
SDMMC_PWREN		USB0.D1 (Set0)	SDIO.PWREN (Set0)	GPIO37	LOANIO37
SDMMC_D0		USB0.D2 (Set0)	**SDIO.D0 (Set0)**	GPIO38	LOANIO38
SDMMC_D1		USB0.D3 (Set0)	**SDIO.D1 (Set0)**	GPIO39	LOANIO39
SDMMC_D4		USB0.D4 (Set0)	SDIO.D4 (Set0)	**GPIO40**	LOANIO40
SDMMC_D5		USB0.D5 (Set0)	SDIO.D5 (Set0)	GPIO41	LOANIO41
SDMMC_D6		USB0.D6 (Set0)	SDIO.D6 (Set0)	GPIO42	LOANIO42
SDMMC_D7		USB0.D7 (Set0)	SDIO.D7 (Set0)	GPIO43	LOANIO43
HPS_GPIO44		USB0.CLK (Set0)		GPIO44	LOANIO44
SDMMC_CCLK_OUT		USB0.STP (Set0)	**SDIO.CLK (Set0)**	GPIO45	LOANIO45
SDMMC_D2		USB0.DIR (Set0)	**SDIO.D2 (Set0)**	GPIO46	LOANIO46
SDMMC_D3		USB0.NXT (Set0)	**SDIO.D3 (Set0)**	GPIO47	LOANIO47
TRACE_CLK			TRACE.CLK (Set0)	GPIO48	LOANIO48
TRACE_D0	**UART0.RX (Set0)**	SPIS0.CLK (Set0)	TRACE.D0 (Set0)	GPIO49	LOANIO49
TRACE_D1	**UART0.TX (Set0)**	SPIS0.MOSI (Set0)	TRACE.D1 (Set0)	GPIO50	LOANIO50
TRACE_D2	**I2C1.SDA (Set0)**	SPIS0.MISO (Set0)	TRACE.D2 (Set0)	GPIO51	LOANIO51
TRACE_D3	**I2C1.SCL (Set0)**	SPIS0.SS0 (Set0)	TRACE.D3 (Set0)	GPIO52	LOANIO52
TRACE_D4	CAN1.RX (Set0)	SPIS1.CLK (Set0)	TRACE.D4 (Set0)	GPIO53	LOANIO53
TRACE_D5	CAN1.TX (Set0)	SPIS1.MOSI (Set0)	TRACE.D5 (Set0)	**GPIO54**	LOANIO54
TRACE_D6	**I2C0.SDA (Set0)**	SPIS1.SS0 (Set0)	TRACE.D6 (Set0)	GPIO55	LOANIO55
TRACE_D7	**I2C0.SCL (Set0)**	SPIS1.MISO (Set0)	TRACE.D7 (Set0)	GPIO56	LOANIO56
SPIM0_CLK	UART0.CTS (Set2) (Set1) (Set0)	I2C1.SDA (Set1)	SPIM0.CLK (Set0)	GPIO57	LOANIO57
SPIM0_MOSI	UART0.RTS (Set2) (Set1) (Set0)	I2C1.SCL (Set1)	SPIM0.MOSI (Set0)	GPIO58	LOANIO58
SPIM0_MISO	UART1.CTS (Set0)	CAN1.RX (Set1)	SPIM0.MISO (Set0)	GPIO59	LOANIO59
SPIM0_SS0	UART1.RTS (Set0)	CAN1.TX (Set1)	SPIM0.SS0 (Set0)	GPIO60	LOANIO60
UART0_RX	SPIM0.SS1 (Set0)	CAN0.RX (Set0)	UART0.RX (Set1)	**GPIO61**	LOANIO61
UART0_TX	SPIM1.SS1 (Set0)	CAN0.TX (Set0)	UART0.TX (Set1)	GPIO62	LOANIO62
I2C0_SDA	**SPIM1.CLK (Set0)**	UART1.RX (Set0)	I2C0.SDA (Set1)	GPIO63	LOANIO63
I2C0_SCL	**SPIM1.MOSI (Set0)**	UART1.TX (Set0)	I2C0.SCL (Set1)	GPIO64	LOANIO64
CAN0_RX	**SPIM1.MISO (Set0)**	UART0.RX (Set2)	CAN0.RX (Set1)	GPIO65	LOANIO65
CAN0_TX	**SPIM1.SS0 (Set0)**	UART0.TX (Set2)	CAN0.TX (Set1)	GPIO66	LOANIO66

Figure 2: The peripherals multiplexer view

5 Conclusion

As a conclusion, we think that this lab was a lot about following the guide, which sometimes didn't help us understand what we were doing. This in part the reason why we have made that mistake. Fortunately, it allowed us to question what we had done, and to learn what we were doing wrong. We had to restart the instruction from the beginning (having had a corruption issue with the VM), so the second time we started to remember a lot better what to do, allowing us to complete it quickly without issues.