AARHUS
UNIVERSITY
SCHOOL OF ENGINEERING

PETER HØGH MIKKELSEN
5. JANUARY, 2017

SOPC INTRODUCTION

INTRODUCING SOPC

- System on Programmable Chips comprises of FPGA + CPU

- *CPU is good for*
- *Control logic, Managing network interfaces, UI*

- *FPGA is good for*
- *Massive parallel data processing, custom interfaces, low latency interfaces*

- Different setups for different applications
- *Hard- / Firm- / Soft Core setups*

“HARD CORE” PROCESSOR

› Processor implemented in silicon with an FPGA on the same die or bundled.

› **Altera:**
› Cyclone V SoC: ARM Cortex-A9 (1-2x @ 900MHz)
› Stratix 10 SoC: ARM Cortex-A53 (4x @ 1500MHz)

› **Xilinx:**
› Zynq 7000 : ARM Cortex-A9 (1-2x @ 900MHz)
› Ultrascale+ : ARM Cortex-A53 (4x @ 1500MHz)

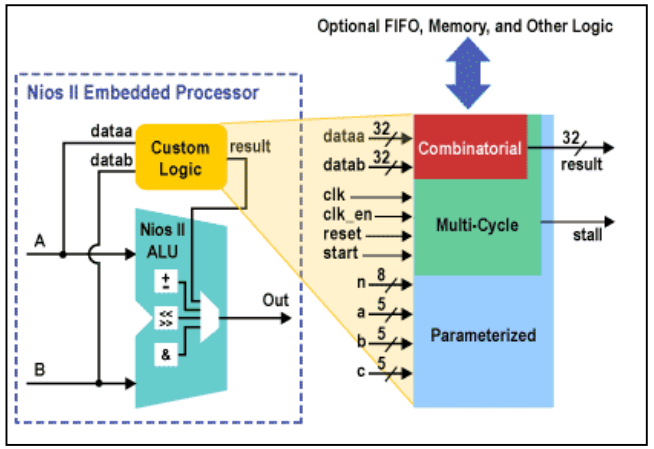
› **Actel:**
› ProASIC3 : ARM Cortex-M1 (1x @ 200MHz)

› *Trend is hard core for high performance + low power*

FIRM CORE PROCESSOR

› Processor created from optimized programmable blocks and structures in an FPGA

› Altera: *Nios II* and Xilinx: *MicroBlaze*
› Very feature-rich and flexible
› Can change following things compile time:
› ALU functionality,
› Number and types of peripherals,
› Memory width, and address space
› Size of instruction/data cache
› Level of debug logic
› Custom Instructions
› Hardware acceleration C2H



› *Lower performance than Hard Cores*

SOFT CORE PROCESSORS

- Processor purely based on general IP, can fit into most FPGAs

- ESA: Leon II (MIPS based)
- Opencores.org:
- 8051, MSP430, AVR….. And many more

- Not architecture dependent
- Higher power and resource consumption.
- Software toolchain and support is not very mature

- *Used for low complexity, low-cost high-volume projects that will evolve into asics*

SOPC TOOLS

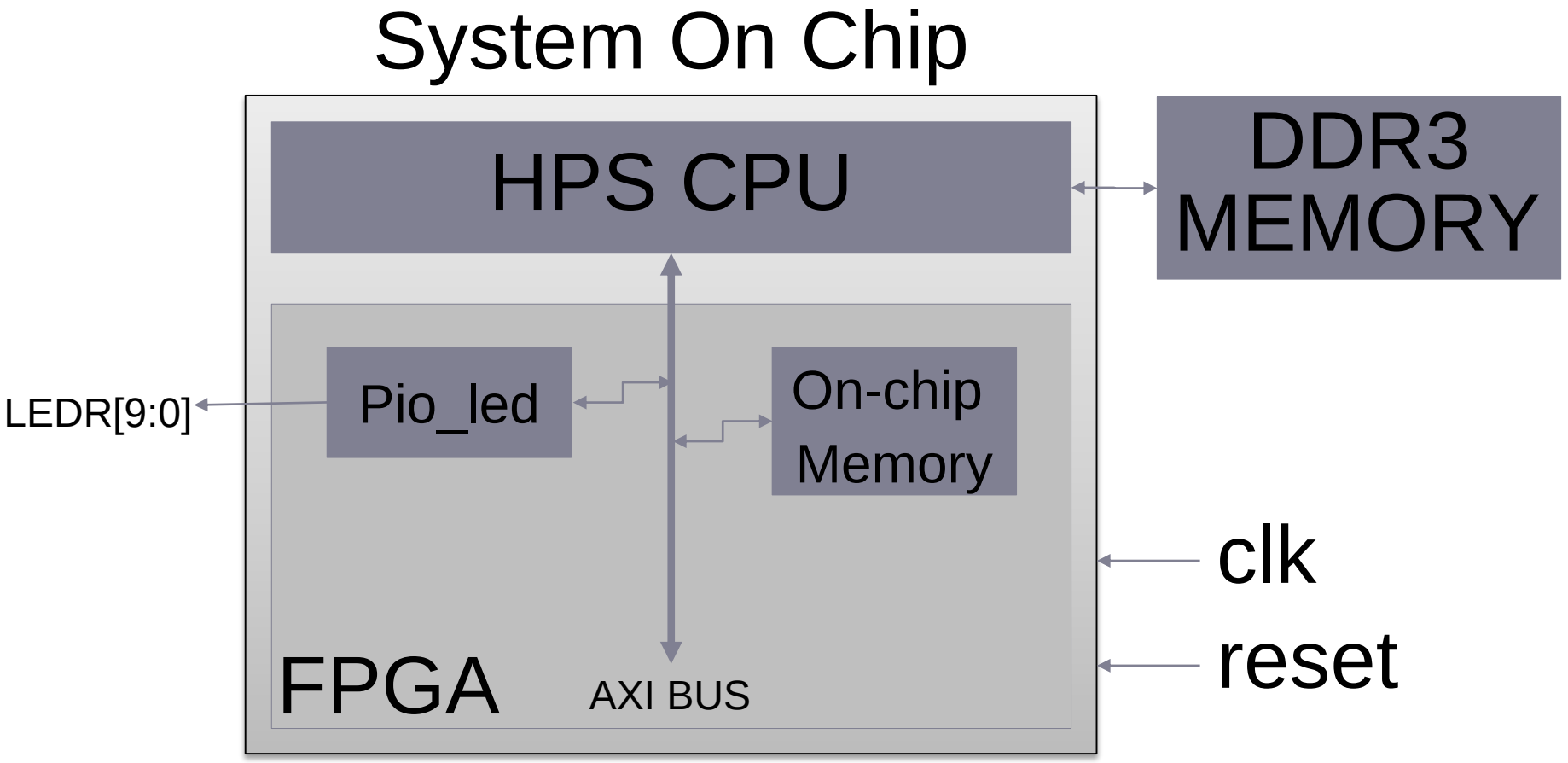
› A specific EDA* tool from every vendor to specify the SOPC system.

› The EDA generates a synthesizable HDL-component for the softcore. Source code is often scrambled – but we got a licenses that unlocks the code.

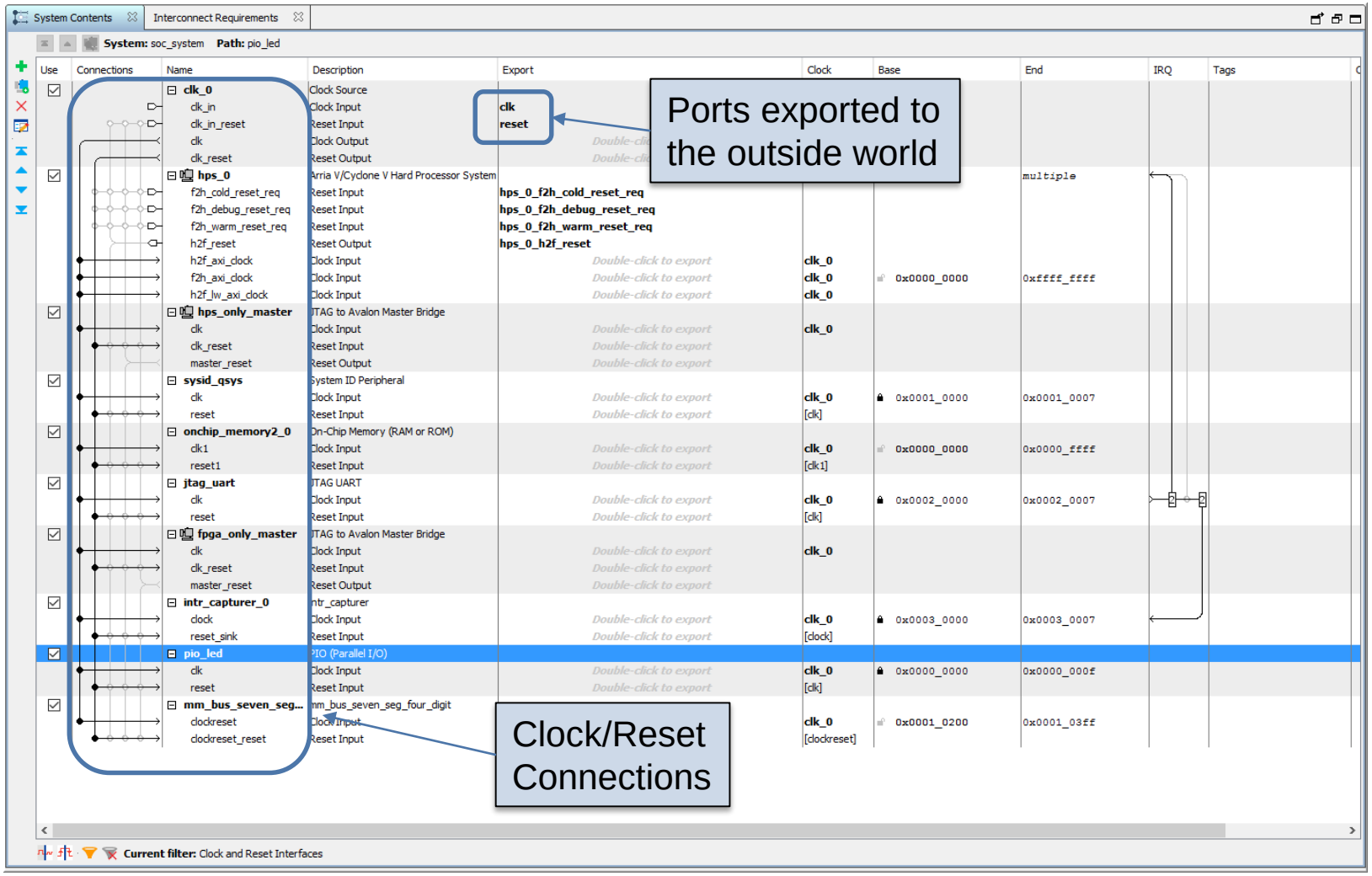
› We can instantiate the HDL-component along with user specific HDL code.

› The Vendor normally supplies C/C++ tool chain for the Softcore.

SOC SYSTEM OVERVIEW

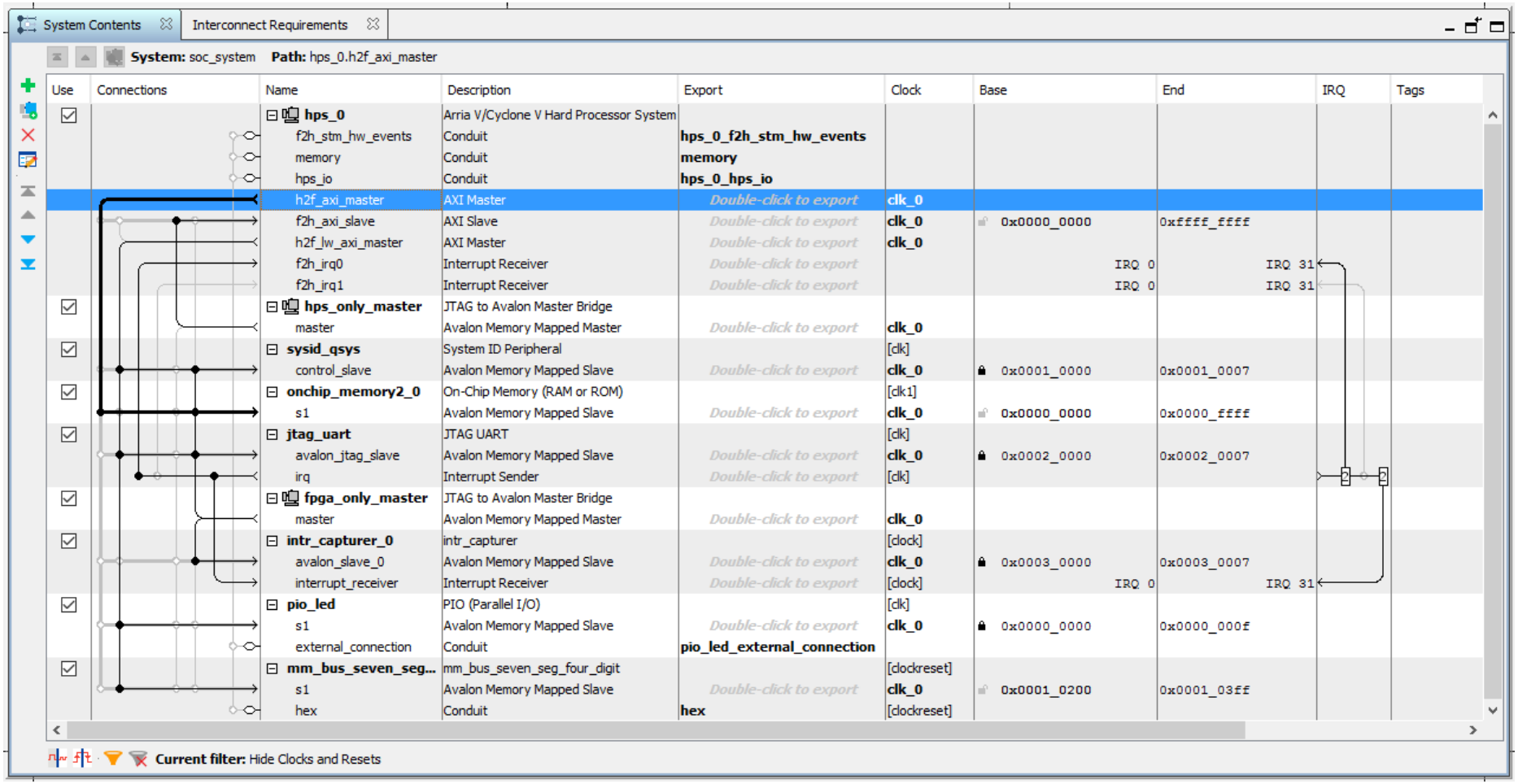


CLOCKS & RESET



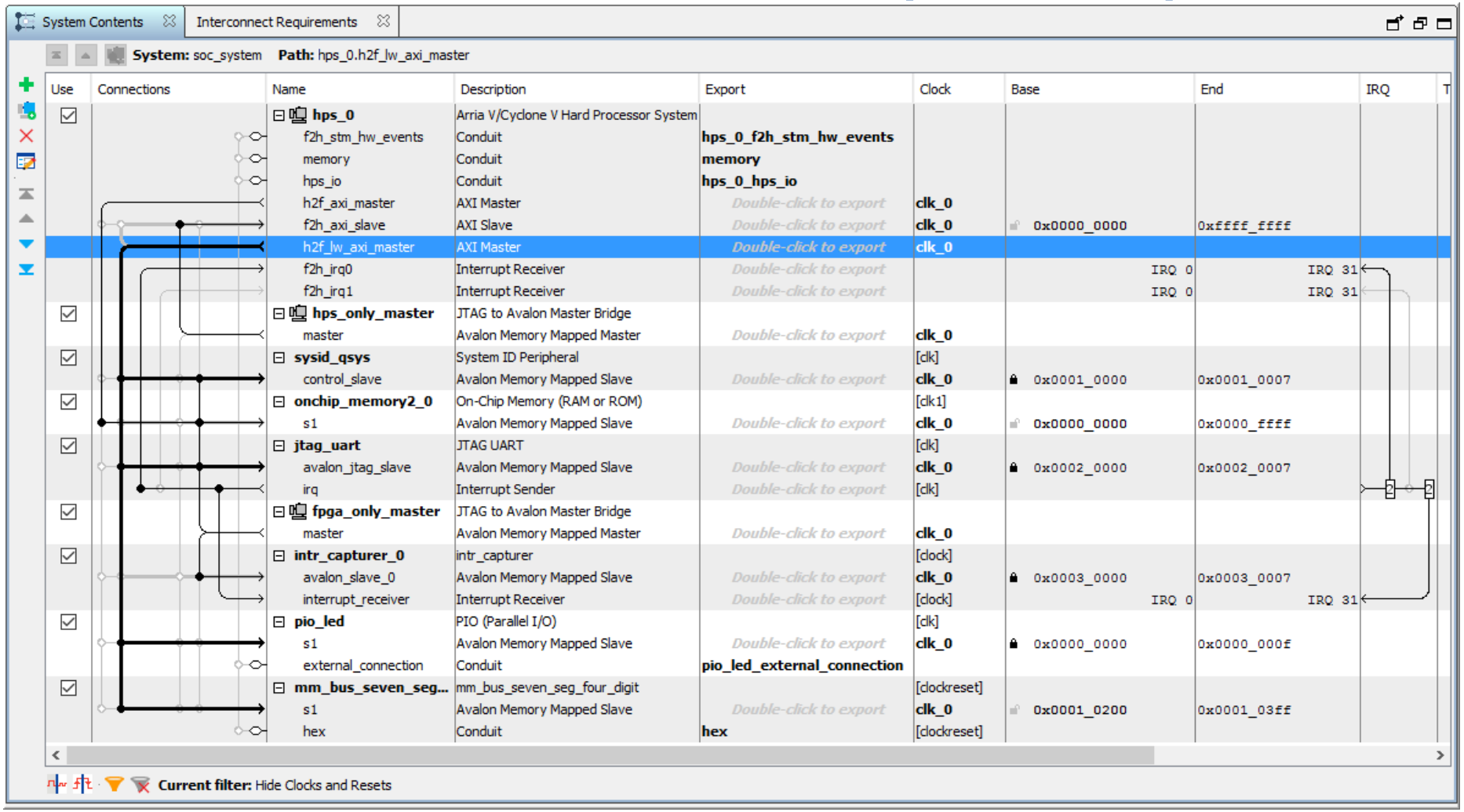
Qsys Clock/Reset View

BUSSES - AXI (64/128-BIT)

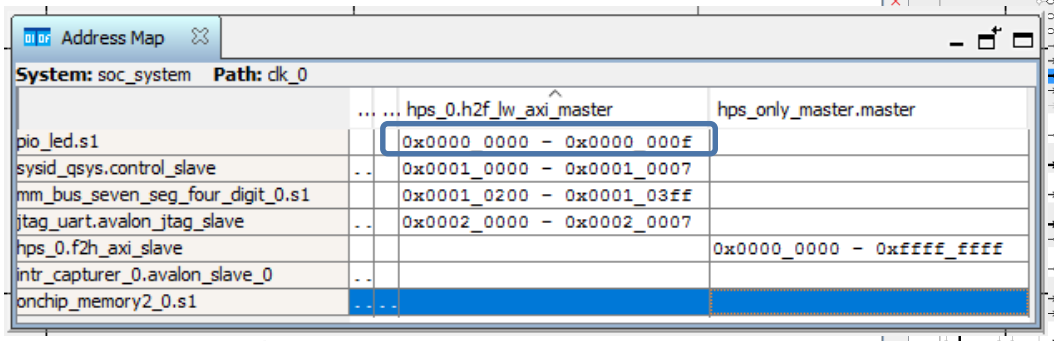


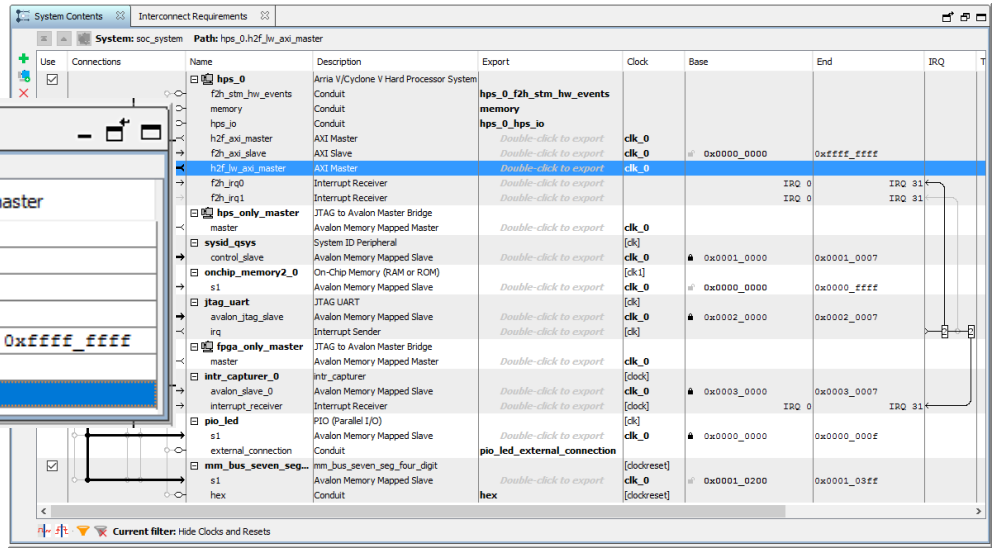
Qsys

BUSSES – AXI LIGHT (32-BIT)



BUSSES – ADDRESSING



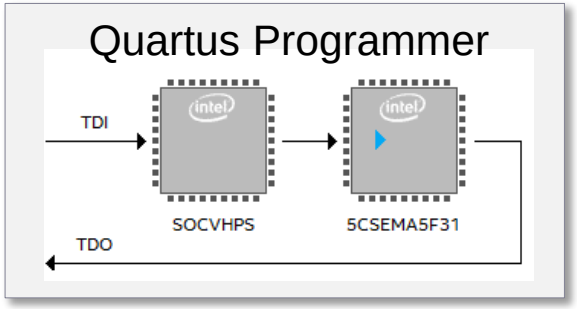


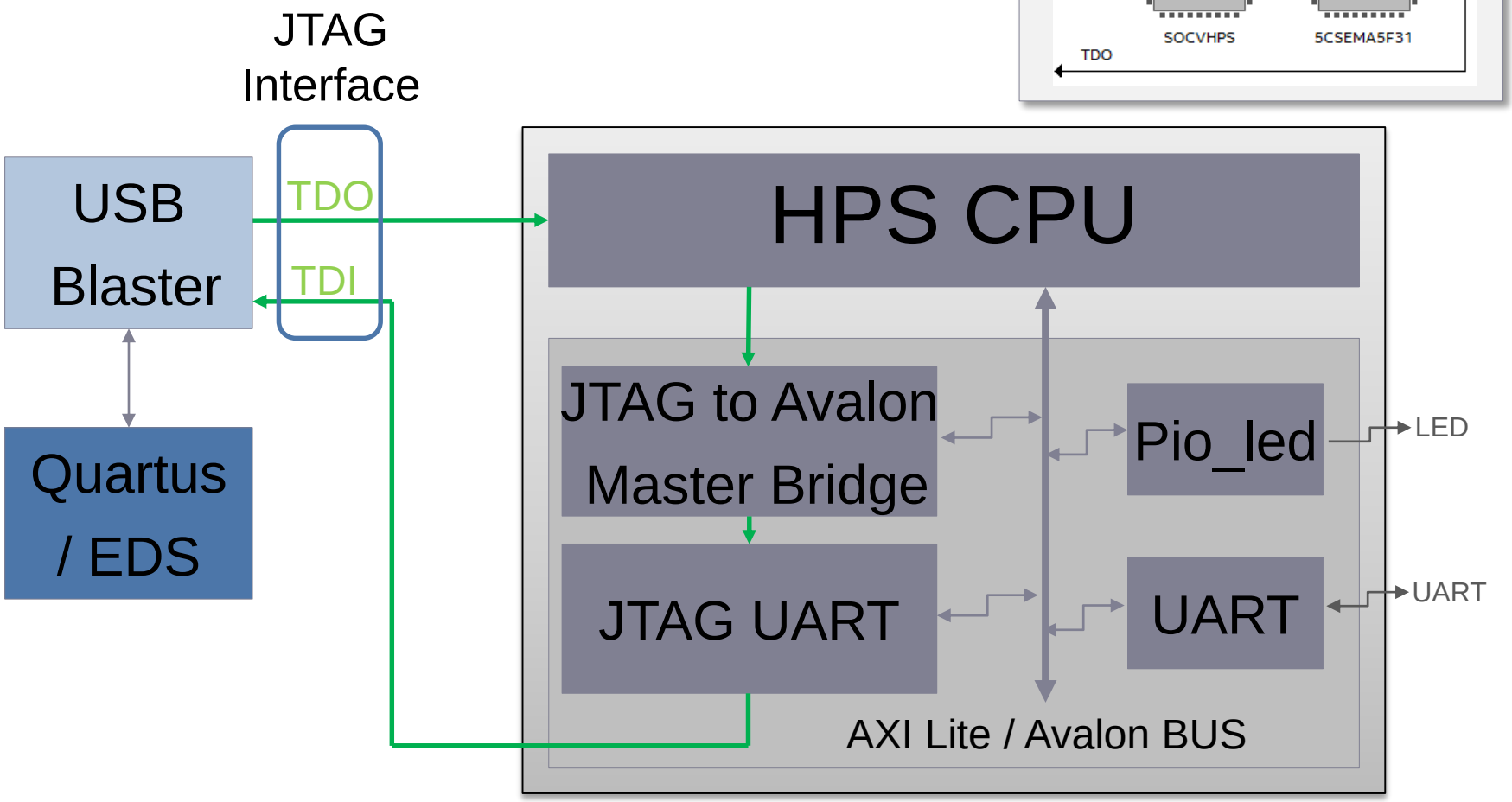
Generate HDL

./generate_hps_qsys_header.sh

```
//hps_0.h
#define PIO_LED_COMPONENT_TYPE
altera_avalon_pio
#define PIO_LED_COMPONENT_NAME pio_led
#define PIO_LED_BASE 0x0
#define PIO_LED_END 0xf
#define PIO_LED_SPAN 16
```

JTAG

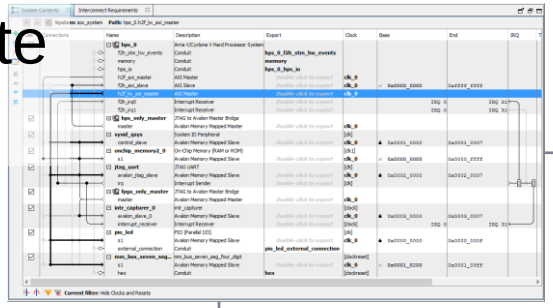




› JTAG is a shift-register interface for programming, test and debugging

WORKFLOW

Generate HDL



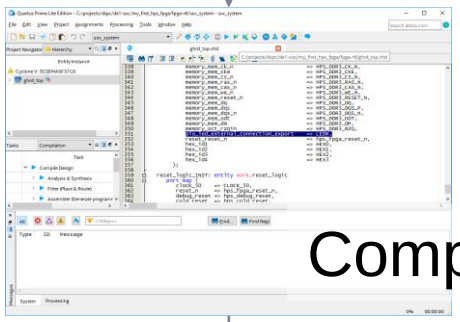
```
-- main.vhd
begin
 u0 : component soc_system
   port map (
     clk_clk        => CLOCK_50,
     pio_led_export => LEDR, …)
```

Instantiate

```
D22502+phm@D22502 /cygdrive/c/projects/dspc/d
$ ./generate_hps_qsys_header.sh
swinfo2header: Creating macro file 'hps_0.h'
```

```
//hps_0.h
#define PIO_LED_BASE 0x0
#define PIO_LED_END 0xf
#define PIO_LED_SPAN 16
```

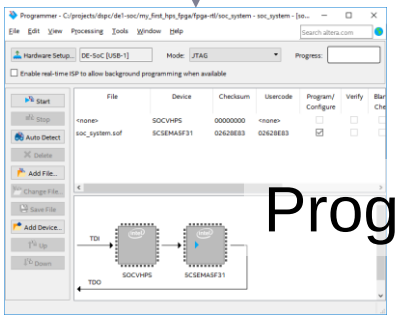
```
#include "hps_0.h"
int main() {
vbase = mmap( NULL,…);
led_addr=vbase\
+ ALT_LWFPGASLVS_OFST\
+ PIO_LED_BASE + …;
*led_addr = 0x55;
```



Compile

```
D22502+phm@D22502 /cygdriv
$ arm-altera-eabi-gcc
```

```
D22502+phm@D22502 /cygdrive/c/pr
$ scp hello root@192.168.10.2:_
```



Program