High Performance Computing

01. Xilinx Vivado HLx Design Suite

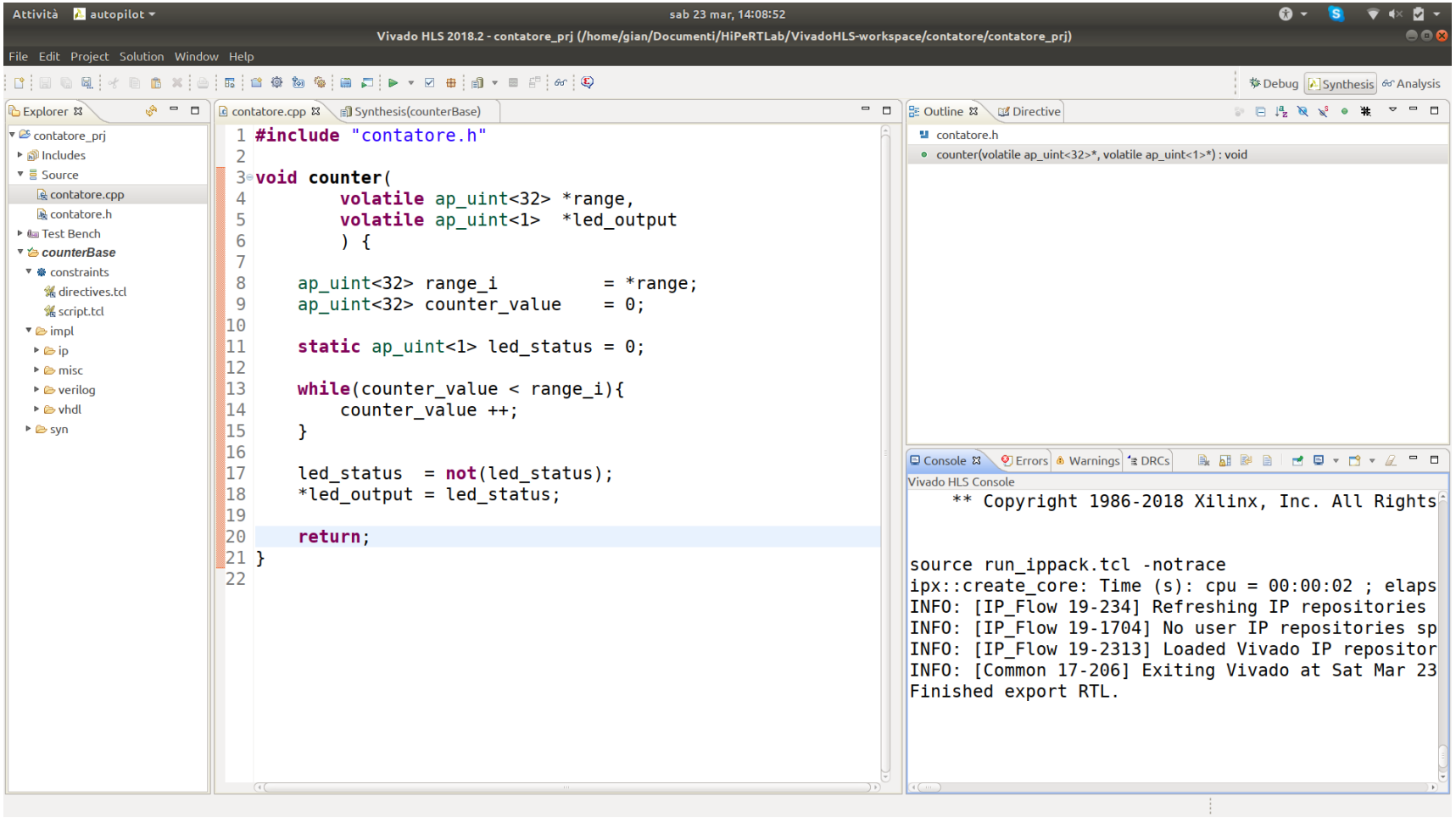
Gianluca Brilli
(Gianluca.Brilli@Unimore.it)
AA 2018-2019

FPGA Design Overview

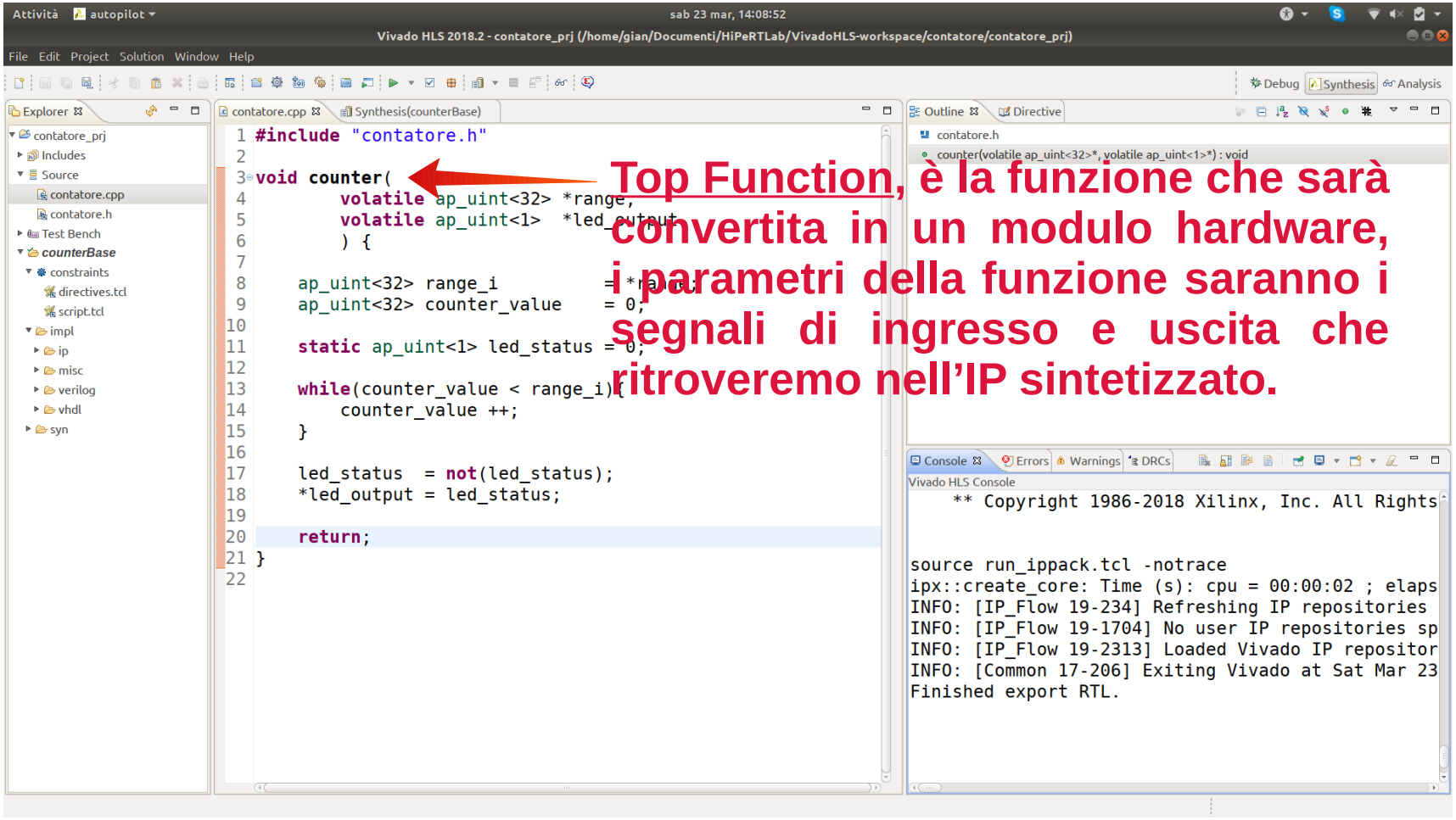
- Strumento principale sviluppato da Xilinx, utilizzato per la programmazione FPGA.
- Suite composta essenzialmente da tre tool:
  - Vivado HLS
  - Vivado HLX
  - Xilinx SDK

Vivado HLS – Interfaccia

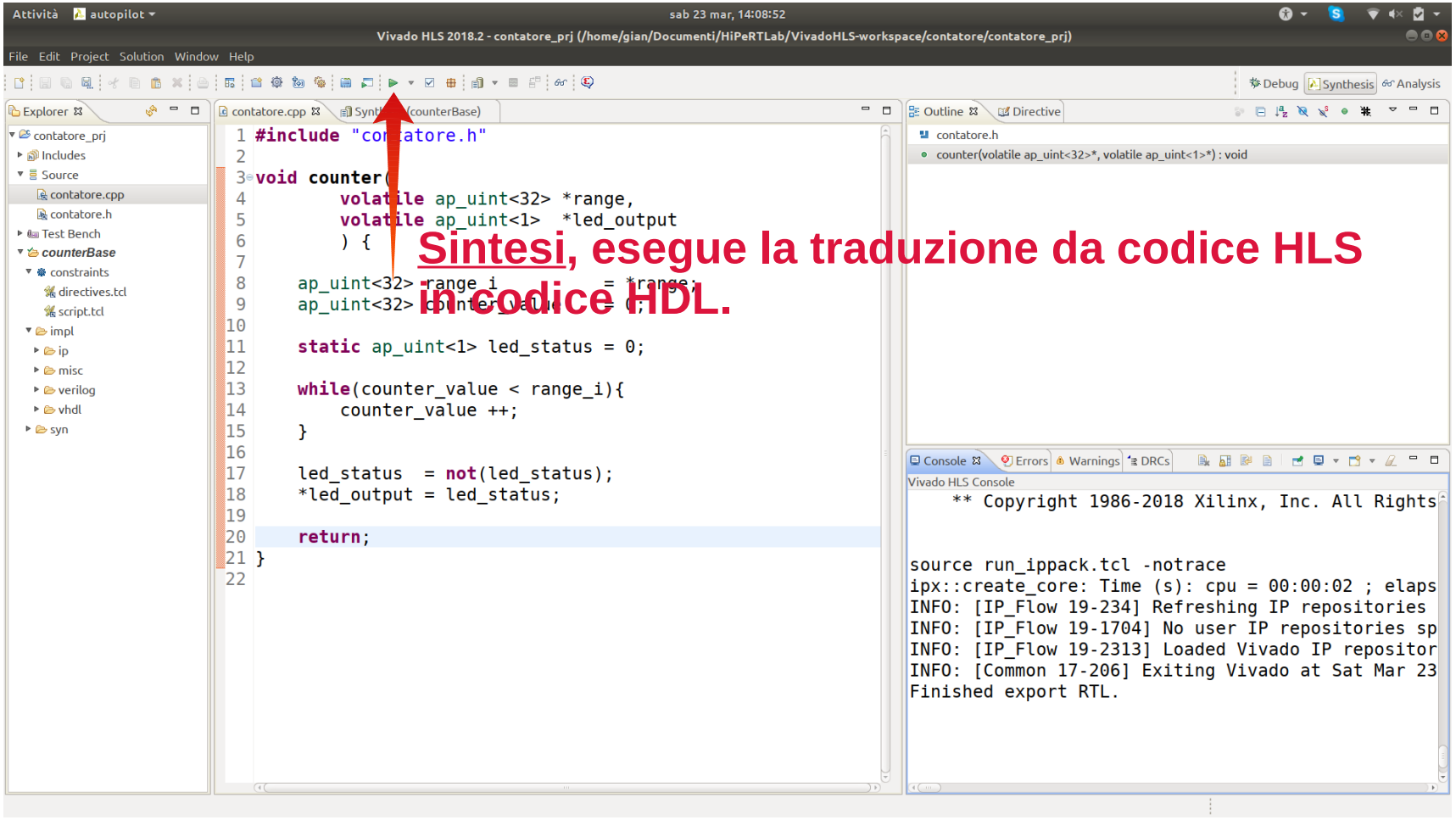
- Una versione di Eclipse modificata.



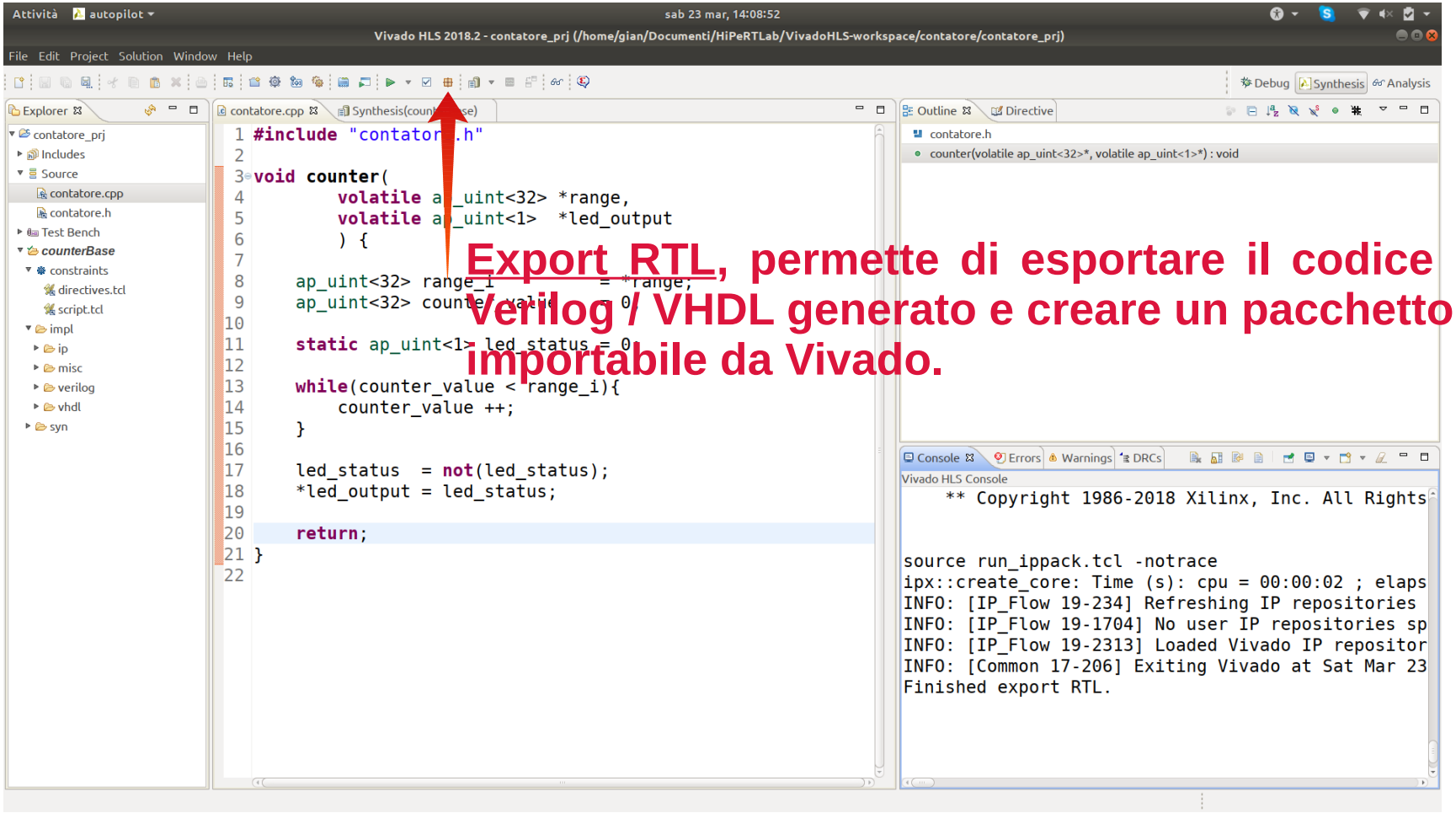
Vivado HLS – Interfaccia



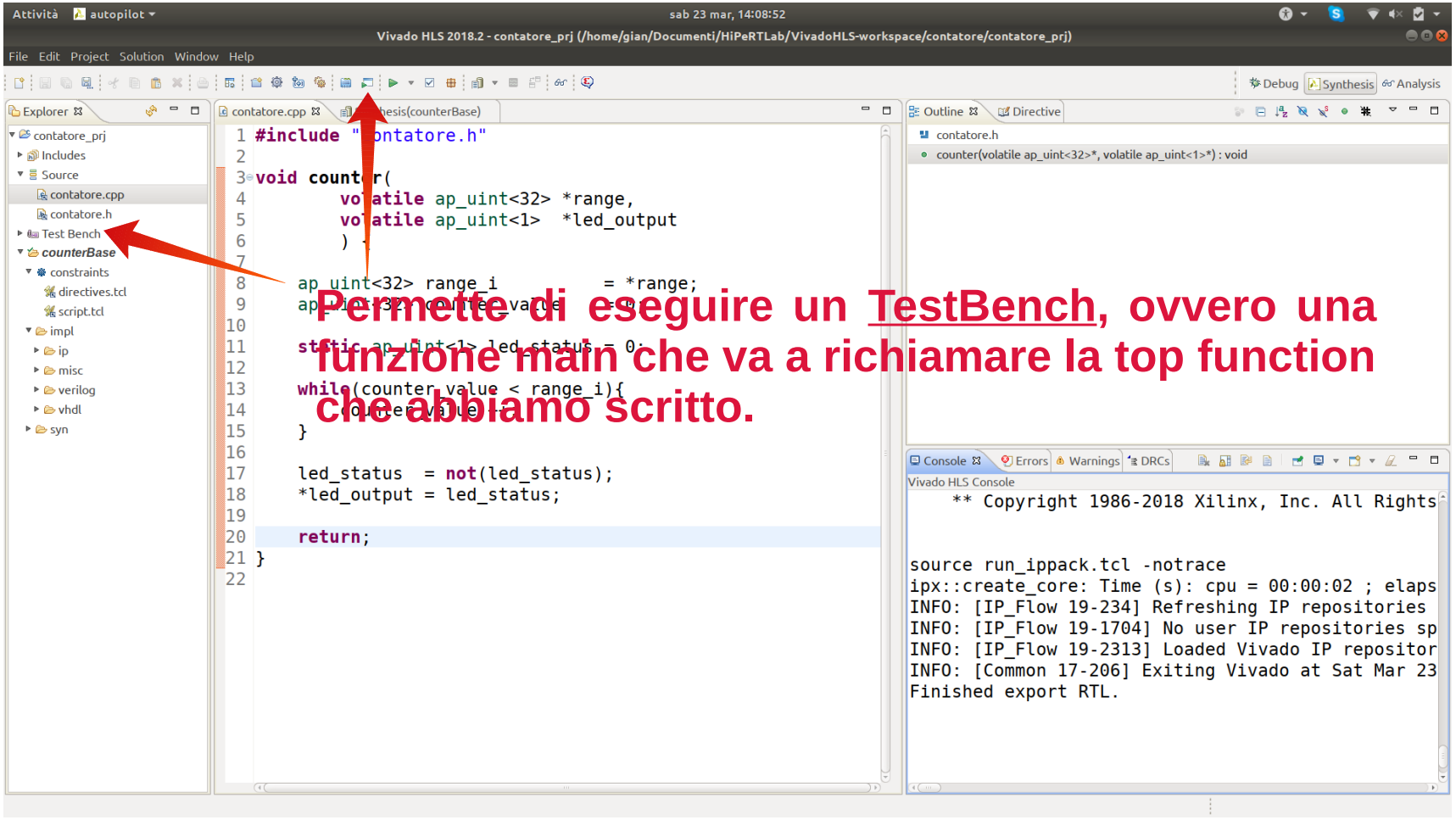
Vivado HLS – Interfaccia



Vivado HLS – Interfaccia



Vivado HLS – Interfaccia



Vivado HLS – Report

General Information

Date:	Thu Mar 7 21:50:46 2019
Version:	2018.2 (Build 2258646 on Thu Jun 14 20:25:20 MDT 2018)
Project:	Convolution_prj
Solution:	solution1
Product family:	zynq
Target device:	xc7z020clg484-1

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.634	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
844870	844870	844870	844870	none

Detail

Instance

Loop

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	9	-	-
Expression	-	-	0	784
FIFO	-	-	-	-
Instance	6	-	1680	2140
Memory	128	-	0	0
Multiplexer	-	-	-	623
Register	0	-	1059	192
Total	134	9	2739	3739
Available	280	220	106400	53200
Utilization (%)	47	4	2	7

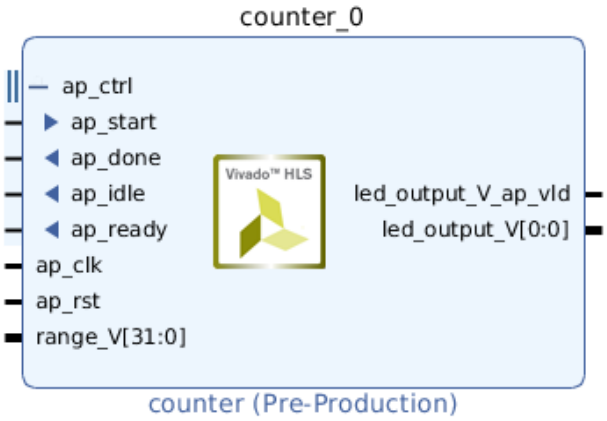
- Generato in automatico dopo la fase di sintesi, ci fornisce alcune informazioni utili sul modulo hardware generato.

Numero di cicli di clock necessari ad eseguire il modulo sintetizzato

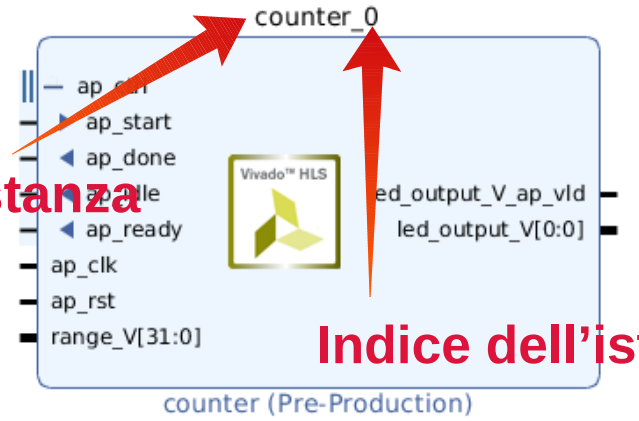
Utilizzo di risorse dell'FPGA, in termini di BRAM, DSP, Flip-Flops e Look-Up-Tables (LUTs)

Vivado HLS – Struttura di un IP

- Un IP generato con Vivado HLS oltre alle porte di ingresso e uscita ha una struttura standard.
- Analizziamo insieme la struttura dell'IP.



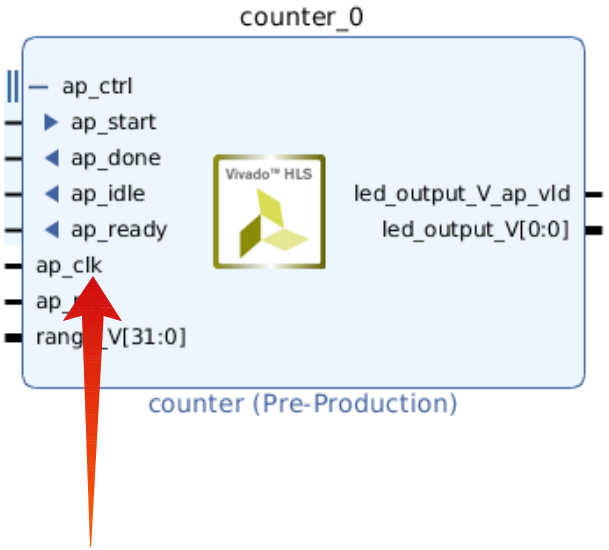
Vivado HLS – Struttura di un IP



Nome dell'istanza dell'IP

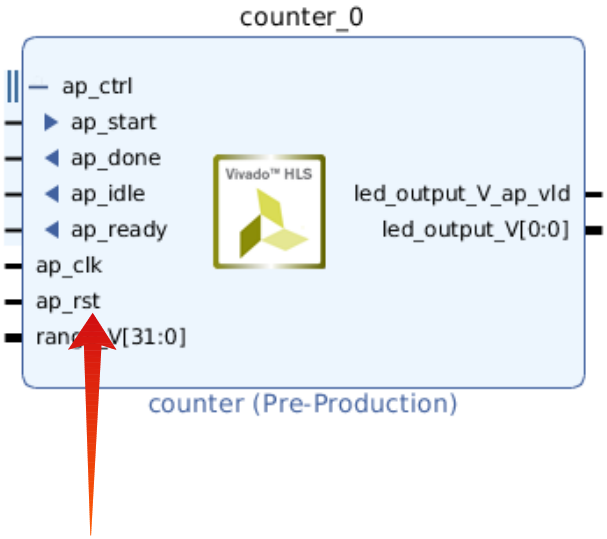
Indice dell'istanza dell'IP

Vivado HLS – Struttura di un IP



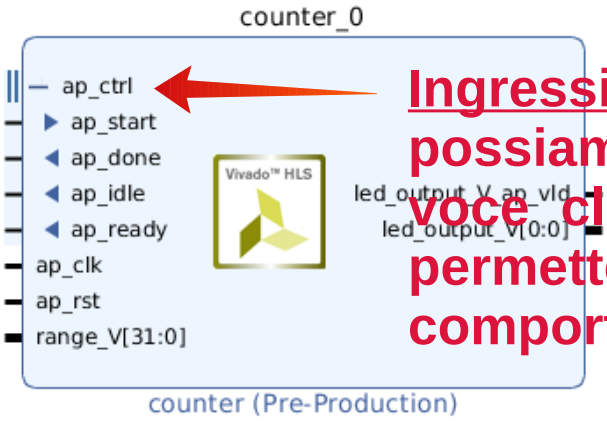
Clock in ingresso, deve essere collegato ad una sorgente esterna. Tipicamente si usa un clock a 100MHz.

Vivado HLS – Struttura di un IP



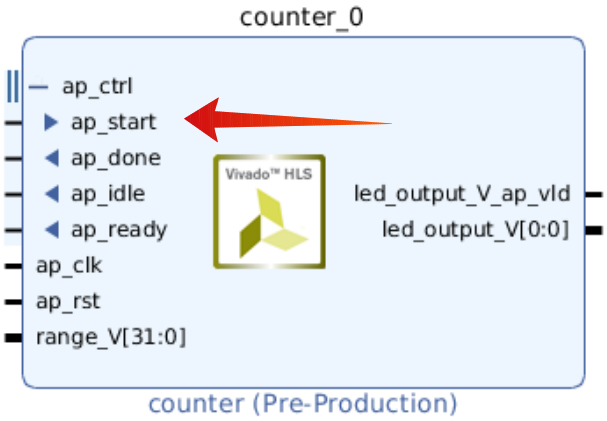
Reset attivo basso, Indispensabile per ogni logica sequenziale porta lo stato del modulo ad un valore noto.

Vivado HLS – Struttura di un IP



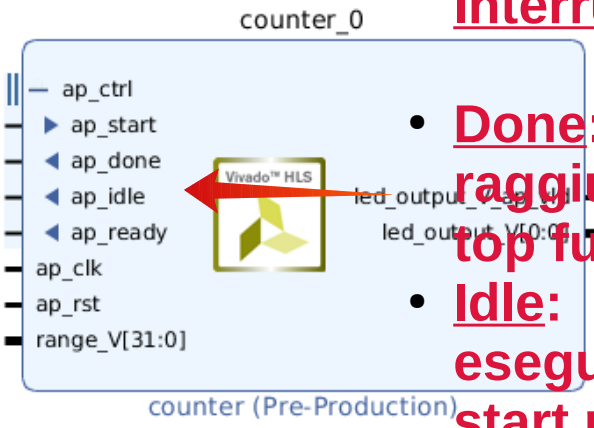
Ingressi di controllo, possiamo espandere la voce cliccando sul "+", permette di controllare il comportamento dell'IP.

Vivado HLS – Struttura di un IP



Bit di start, se settato al livello logico “1”, il modulo inizia ad eseguire.

Vivado HLS – Struttura di un IP

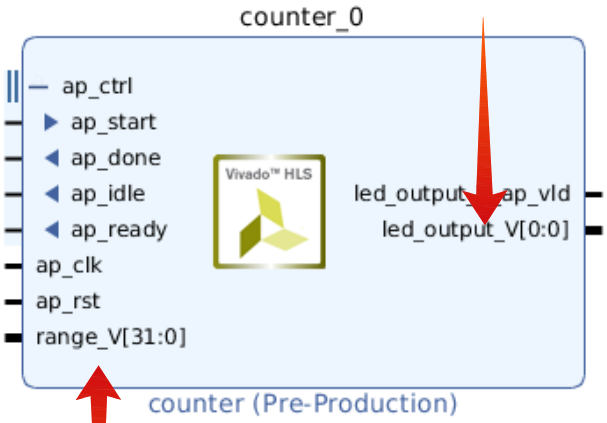


Bit di stato, indicano lo stato dell'IP e possono essere usati come Interrupt.

- **Done: Il modulo ha raggiunto il termine della top function.**
- **Idle: Il modulo non sta eseguendo, (segnale di start non settato).**
- **Ready: Il modulo è pronto per ri eseguire la propria top function.**

Vivado HLS – Struttura di un IP

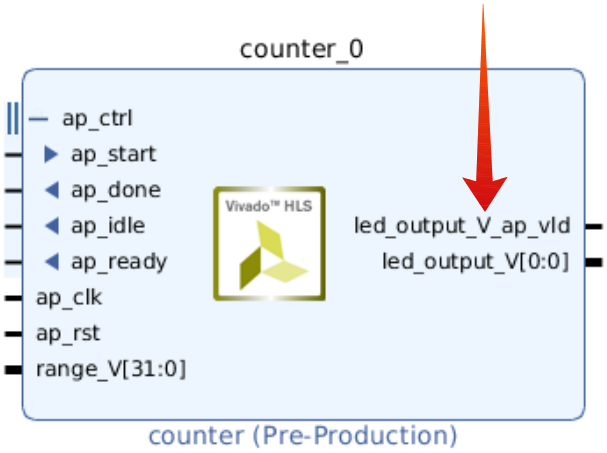
Output, che abbiamo specificato in Vivado HLS.



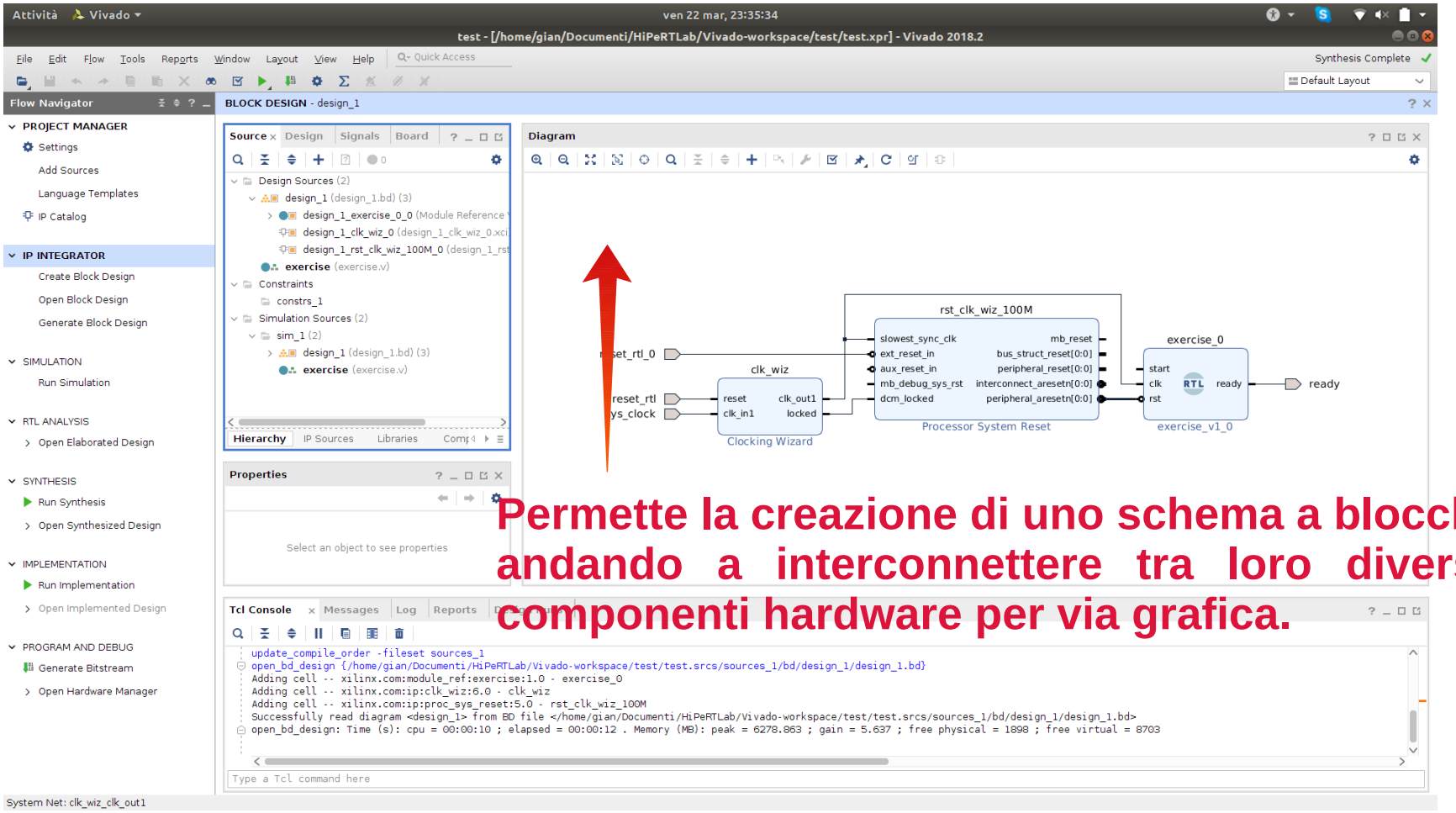
Input, che abbiamo specificato in Vivado HLS.

Vivado HLS – Struttura di un IP

Bit di validità, quando è “1” l’uscita contiene un valore valido (non indispensabile).

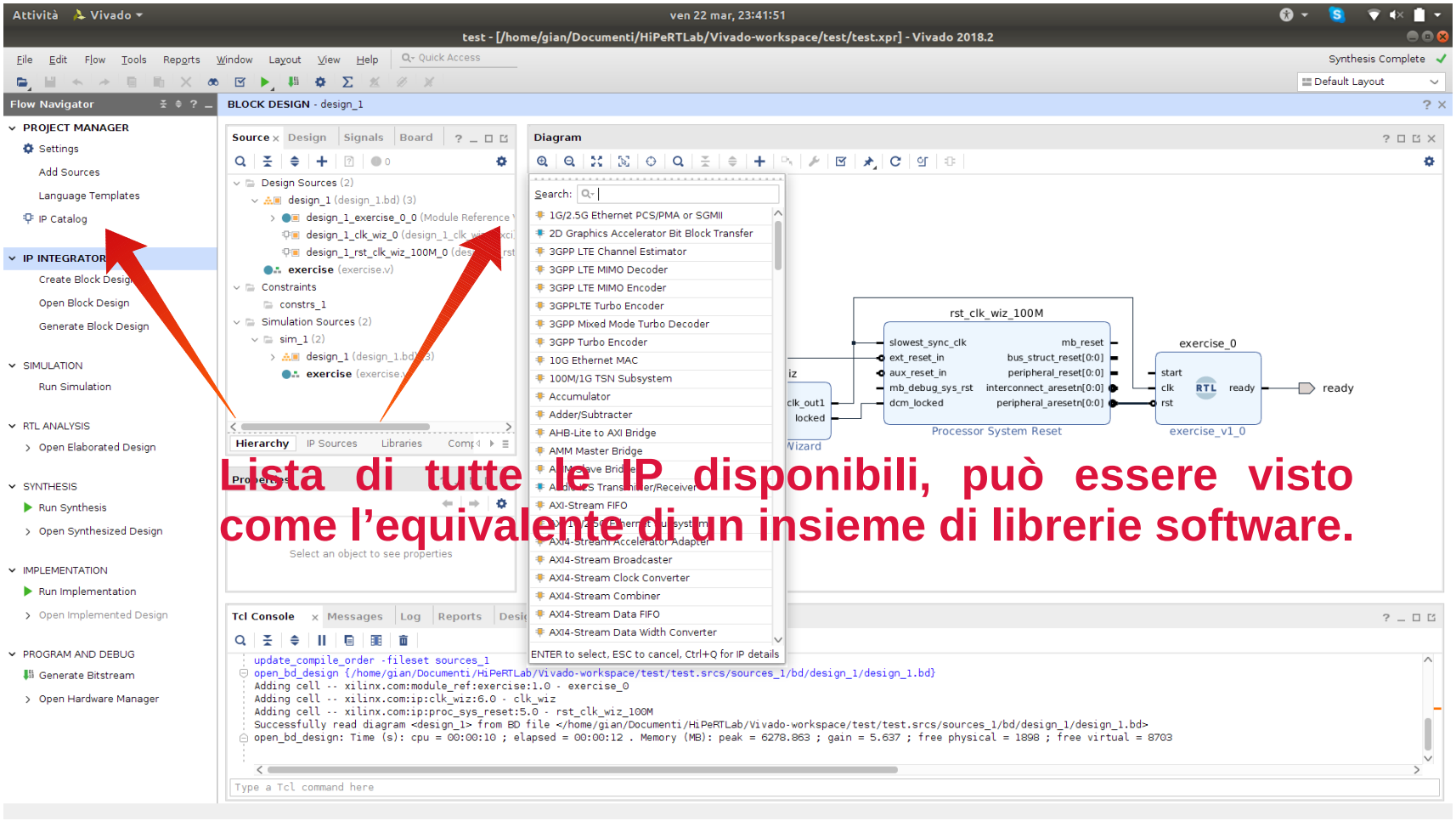


Vivado HLx – Block Design



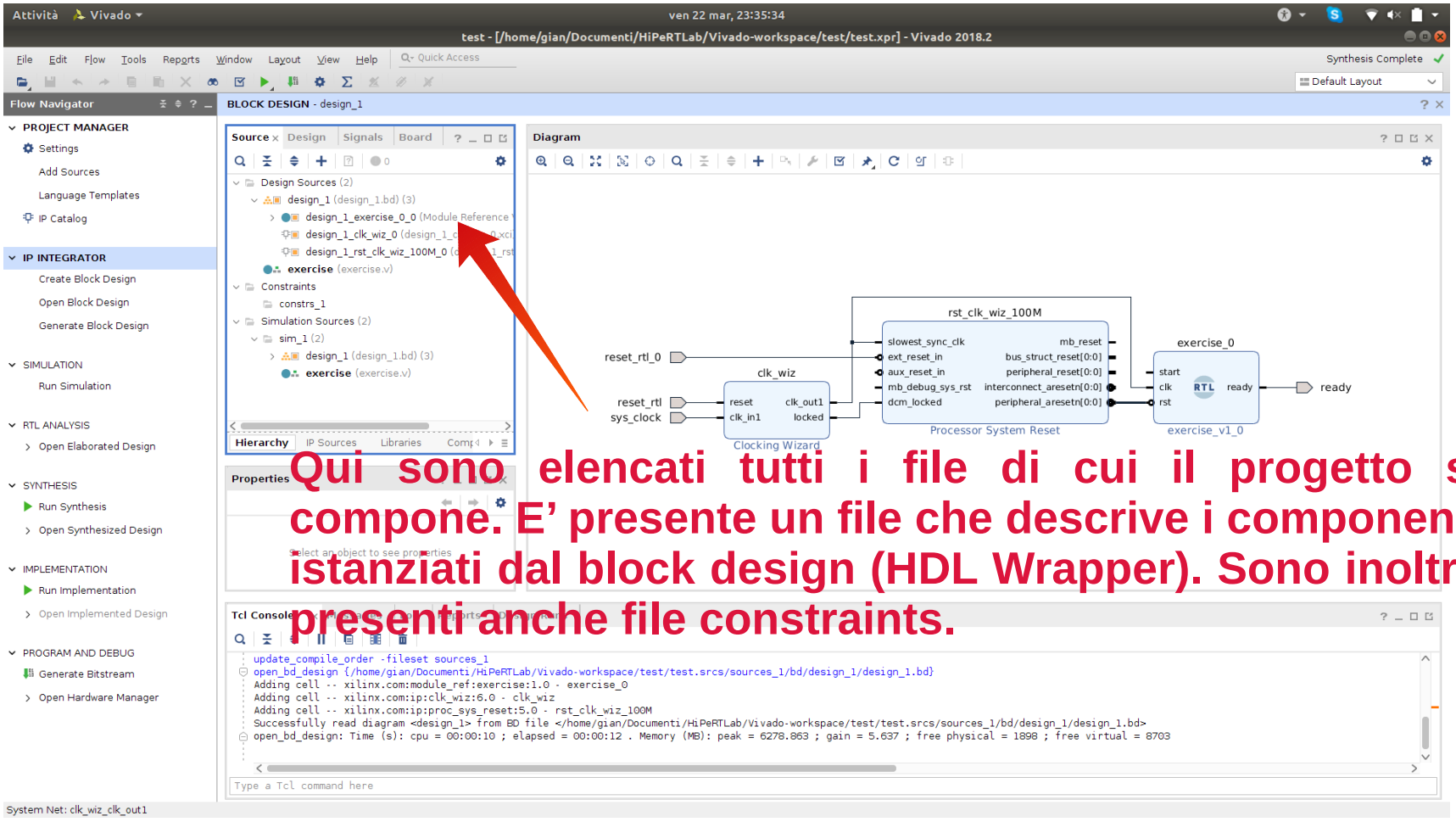
Permette la creazione di uno schema a blocchi andando a interconnettere tra loro diversi componenti hardware per via grafica.

Vivado HLx – IP Catalog



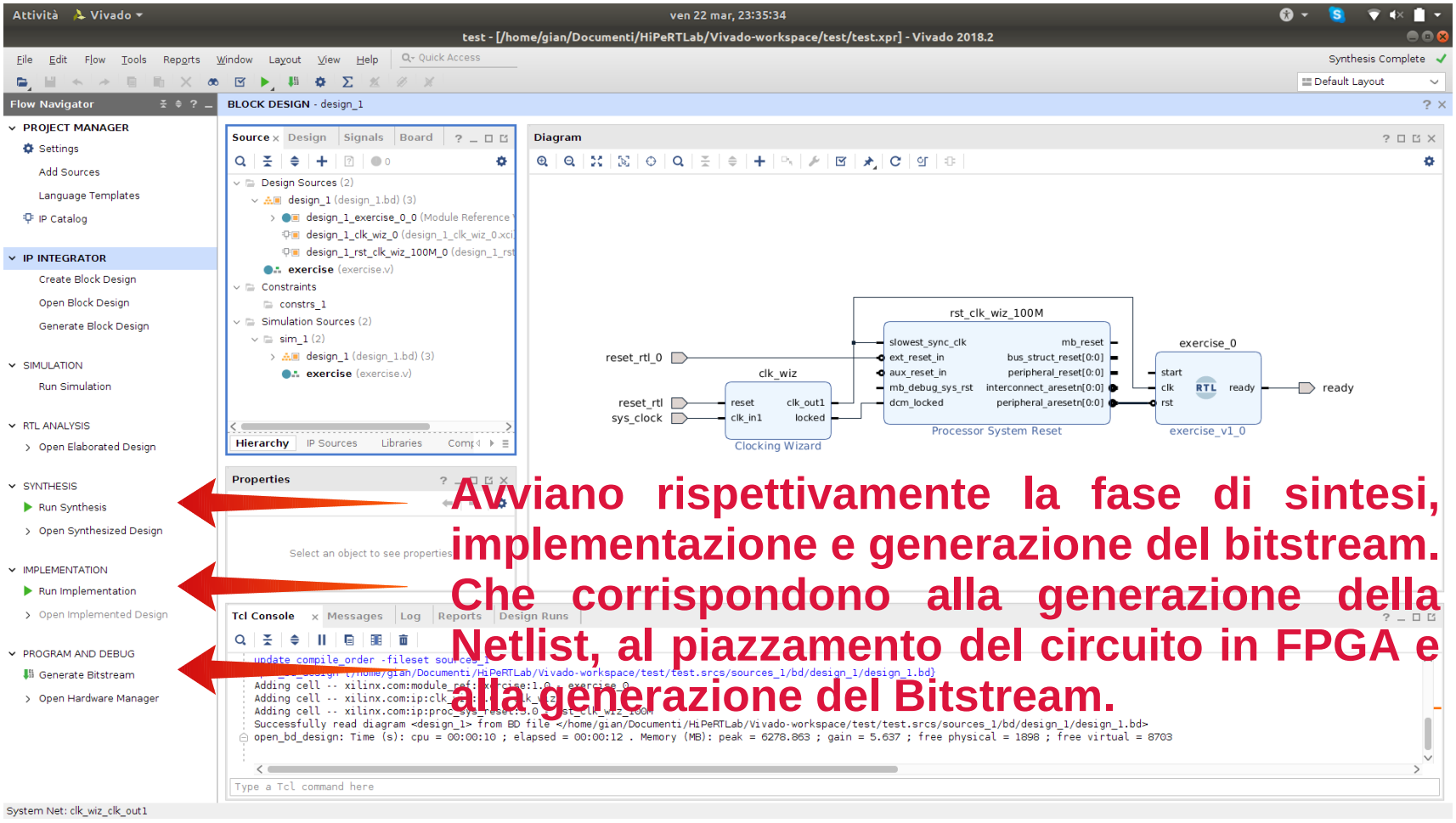
Lista di tutte le IP disponibili, può essere visto come l'equivalente di un insieme di librerie software.

Vivado HLx – Struttura Progetto



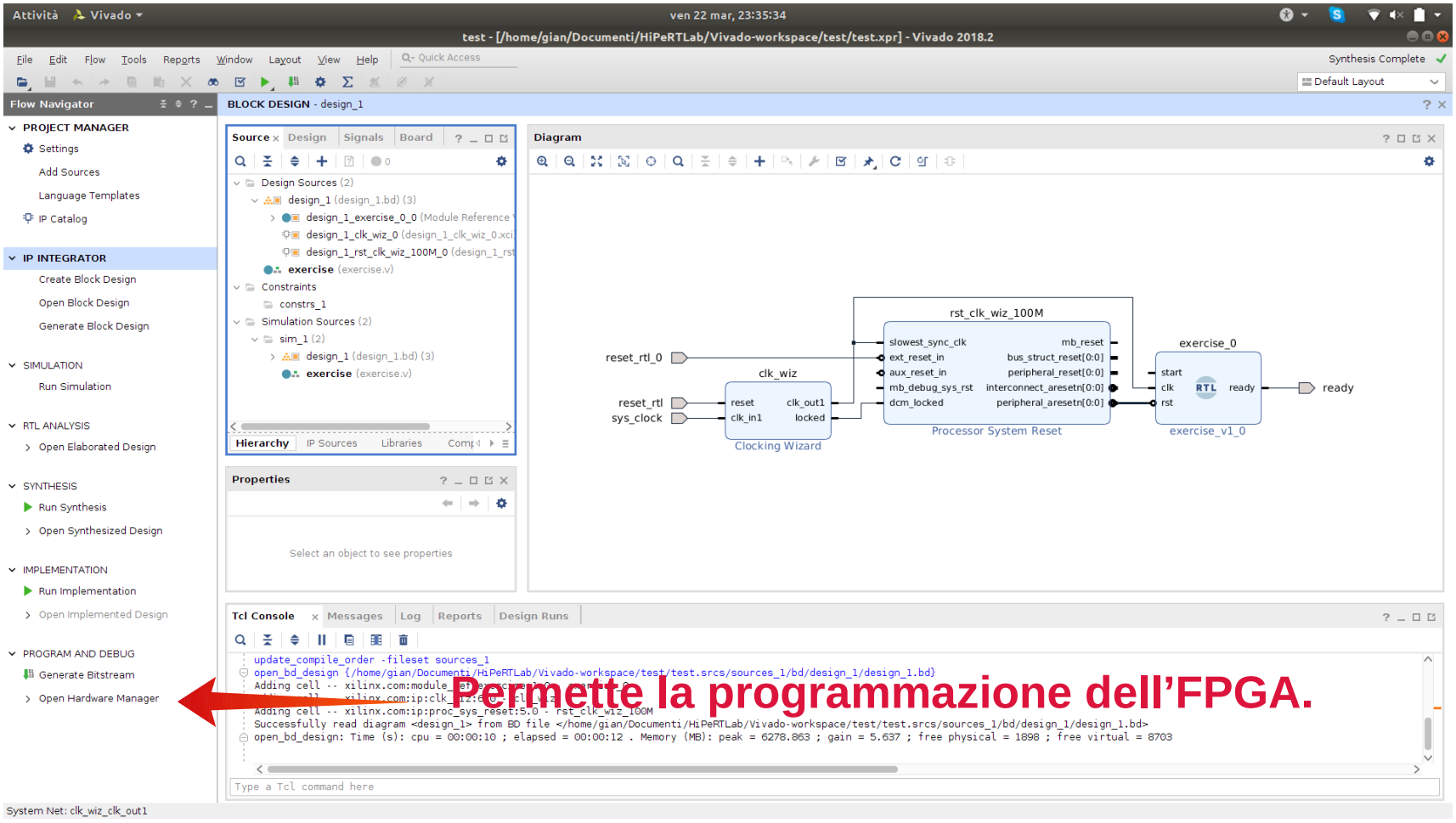
Qui sono elencati tutti i file di cui il progetto si compone. E' presente un file che descrive i componenti istanziati dal block design (HDL Wrapper). Sono inoltre presenti anche file constraints.

Vivado HLx – Sintesi



Avviano rispettivamente la fase di sintesi, implementazione e generazione del bitstream. Che corrispondono alla generazione della Netlist, al piazzamento del circuito in FPGA e alla generazione del Bitstream.

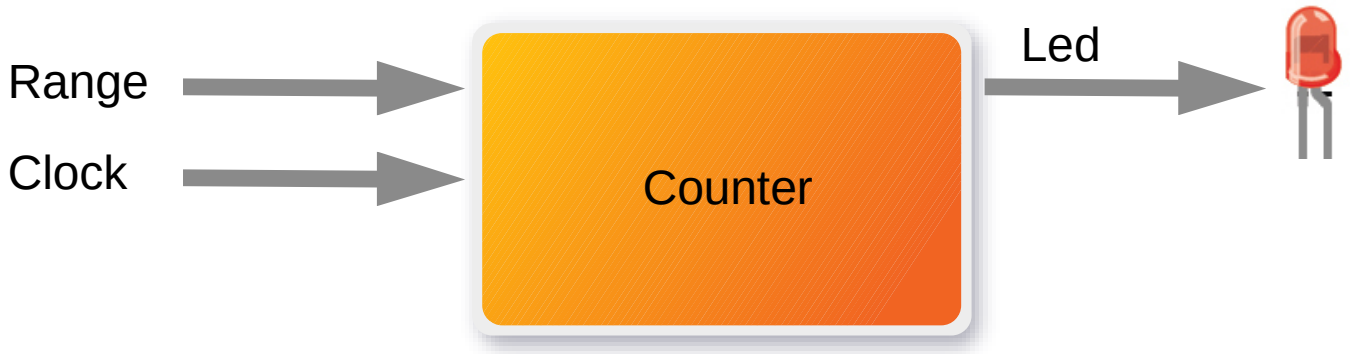
Vivado HLx – Hardware Manager



Permette la programmazione dell'FPGA.

HLS – Esercitazione 01

- Provare a realizzare in FPGA un modulo contatore che dopo N cicli di clock vada ad invertire lo stato di un led, mappato sulla ZedBoard.



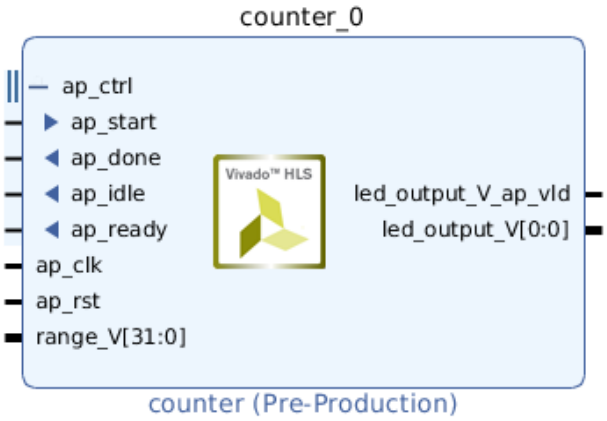
HLS – Esercitazione 01

- Provare a sintetizzare il seguente codice C:

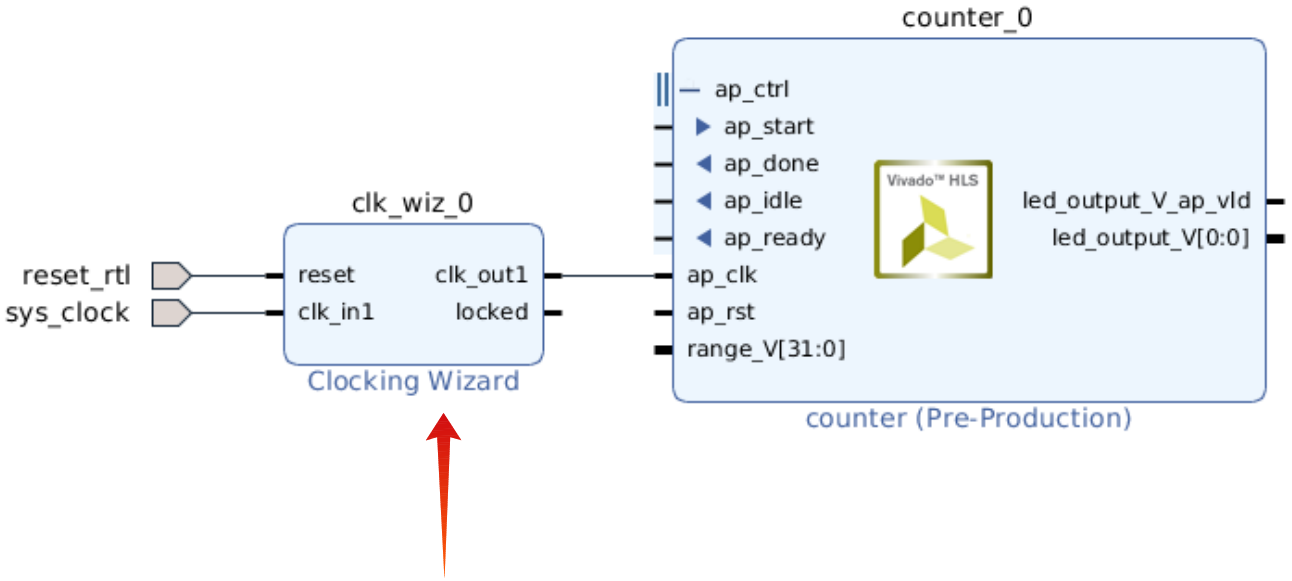
```
1
2   void counter(
3           volatile int   range,
4           volatile bool *led_out) {
5
6       static bool  led_status = 0;
7       volatile int temp_count = 0;
8
9       while (temp_count < range) {
10          temp_count = temp_count + 1;
11      }
12
13      led_status = not(led_status);
14
15      *led_out   = led_status;
16  }
```

Creazione Block Design

- Andiamo a realizzare lo schema a blocchi del nostro sistema a partire dall'IP che abbiamo creato.



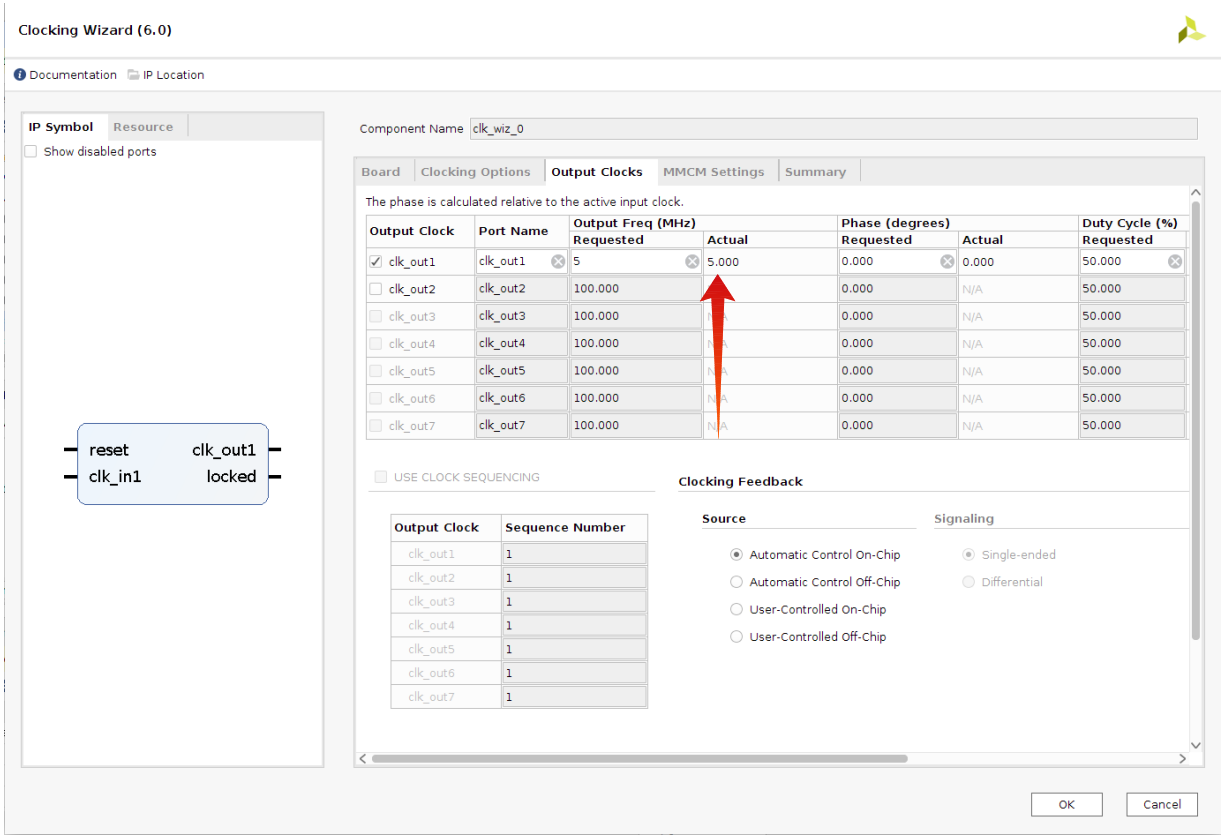
Creazione Block Design - Clock



Clocking Wizard: fornisce un uscita di clock configurabile che possiamo utilizzare nel nostro design.

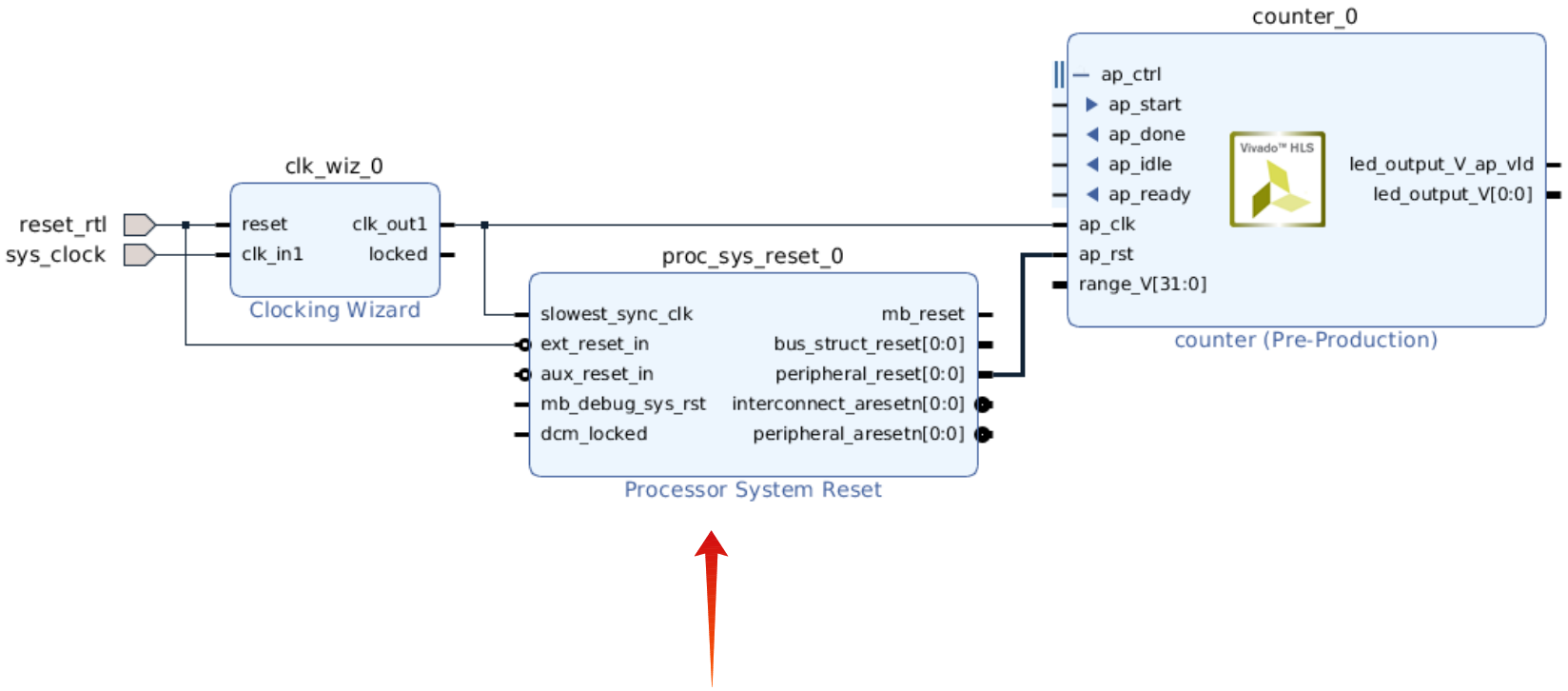
Creazione Block Design - Clock

- Configuriamo il Clocking Wizard, in modo tale che abbia la più bassa frequenza possibile.
- In questo modo vediamo più chiaramente gli effetti sul led.



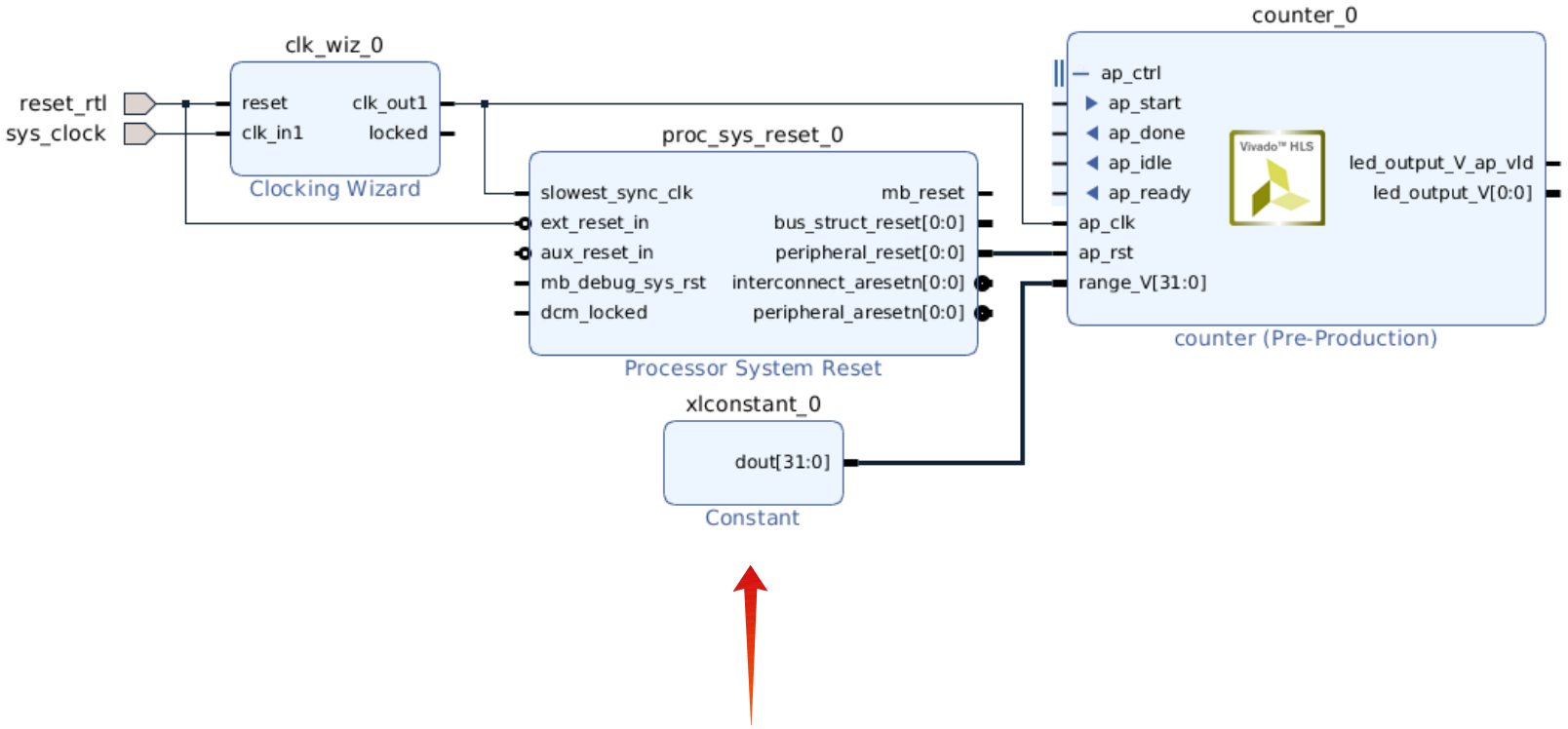
- Impostiamo un clock molto basso, in modo da veder lampeggiare il led.
- Ad esempio 5 MHz.

Creazione Block Design - Reset



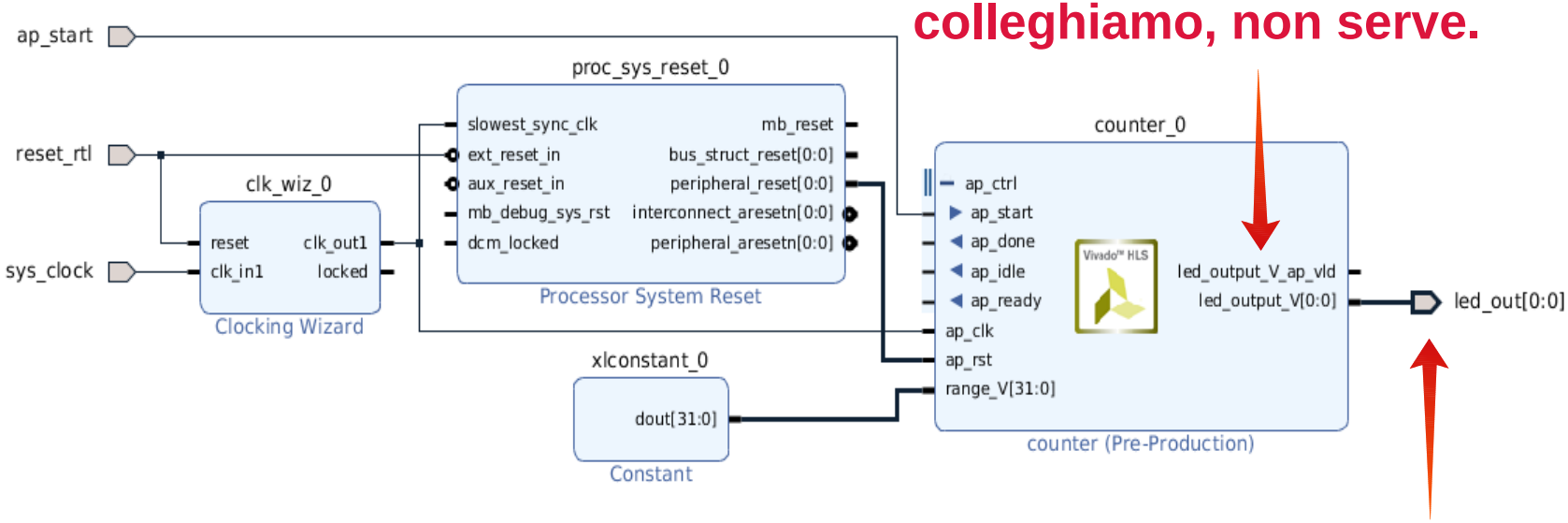
Processor System Reset: gestisce il reset della PS e delle periferiche IP.

Creazione Block Design - Range



<u>Constant</u>: permette di cablare un valore fissato nel design.

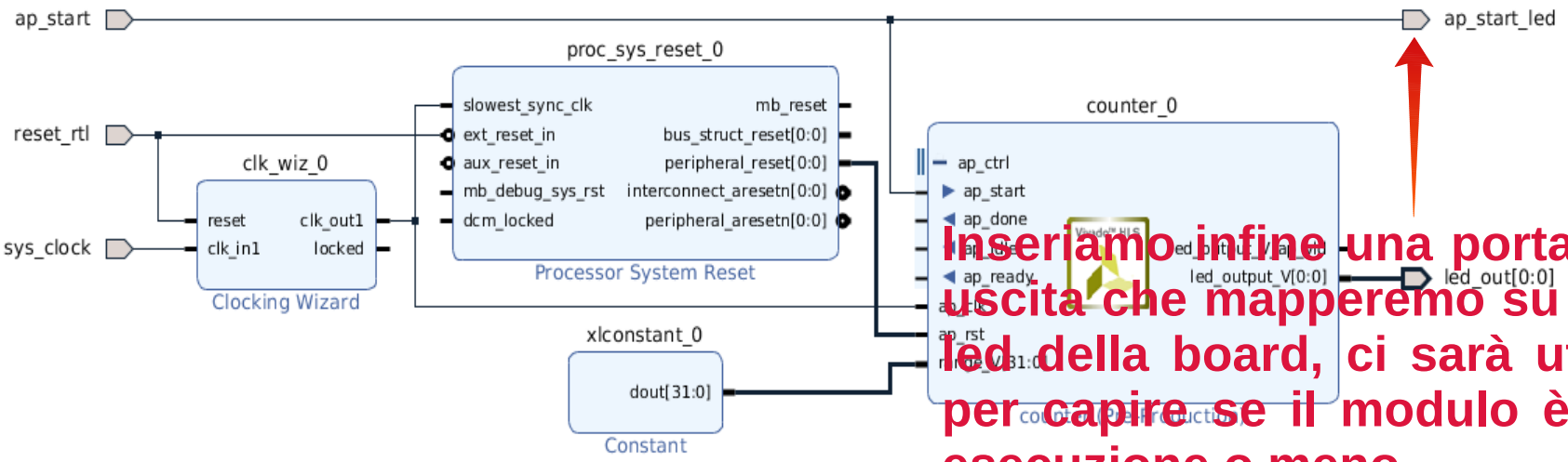
Creazione Block Design - Output



Bit di validità: Non lo colleghiamo, non serve.

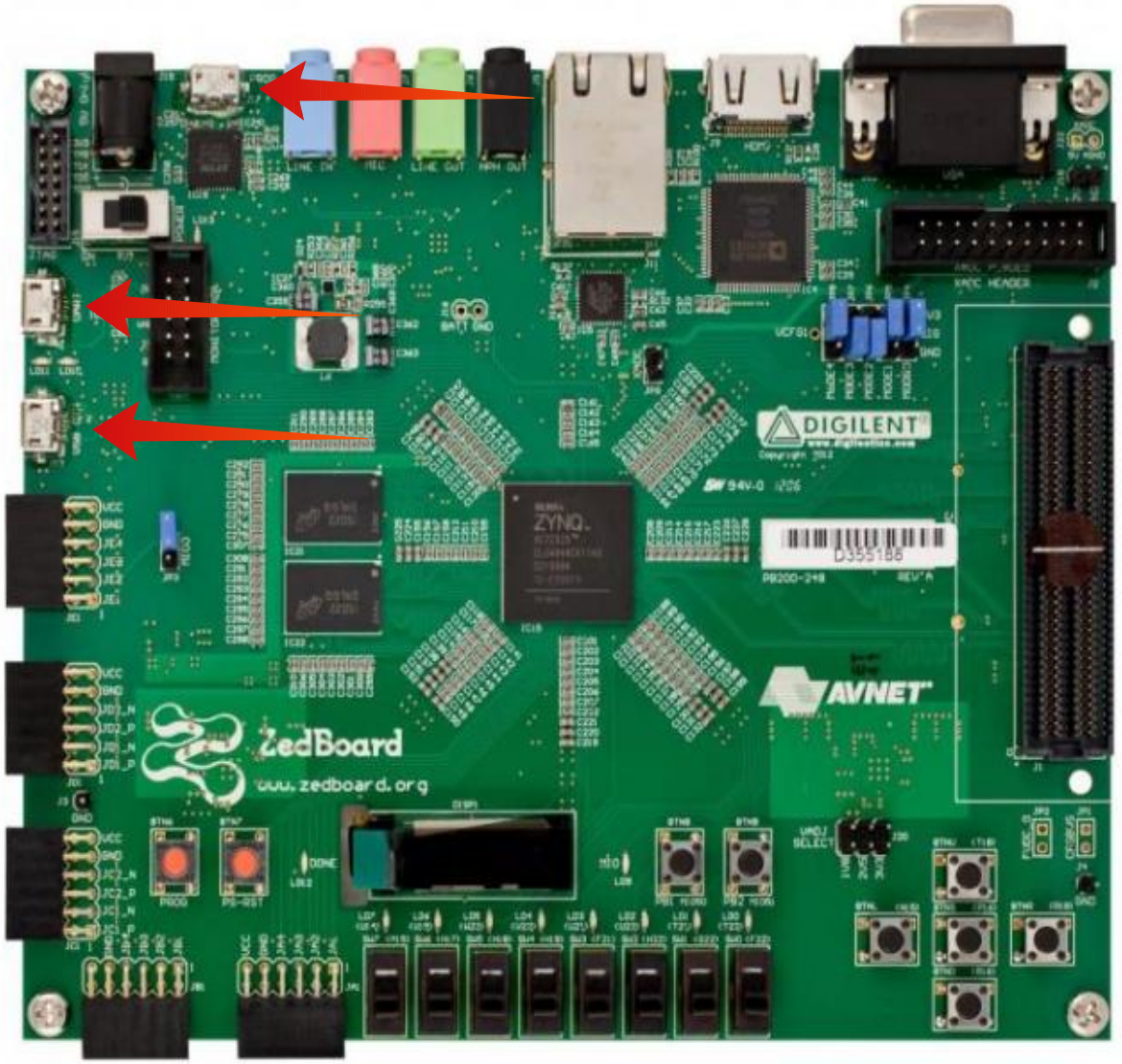
Output: mappiamo l'uscita su un led della ZedBoard.

Creazione Block Design - Output



Inseriamo infine una porta di uscita che mapperemo su un led della board, ci sarà utile per capire se il modulo è in esecuzione o meno.

ZedBoard – Interfacce USB

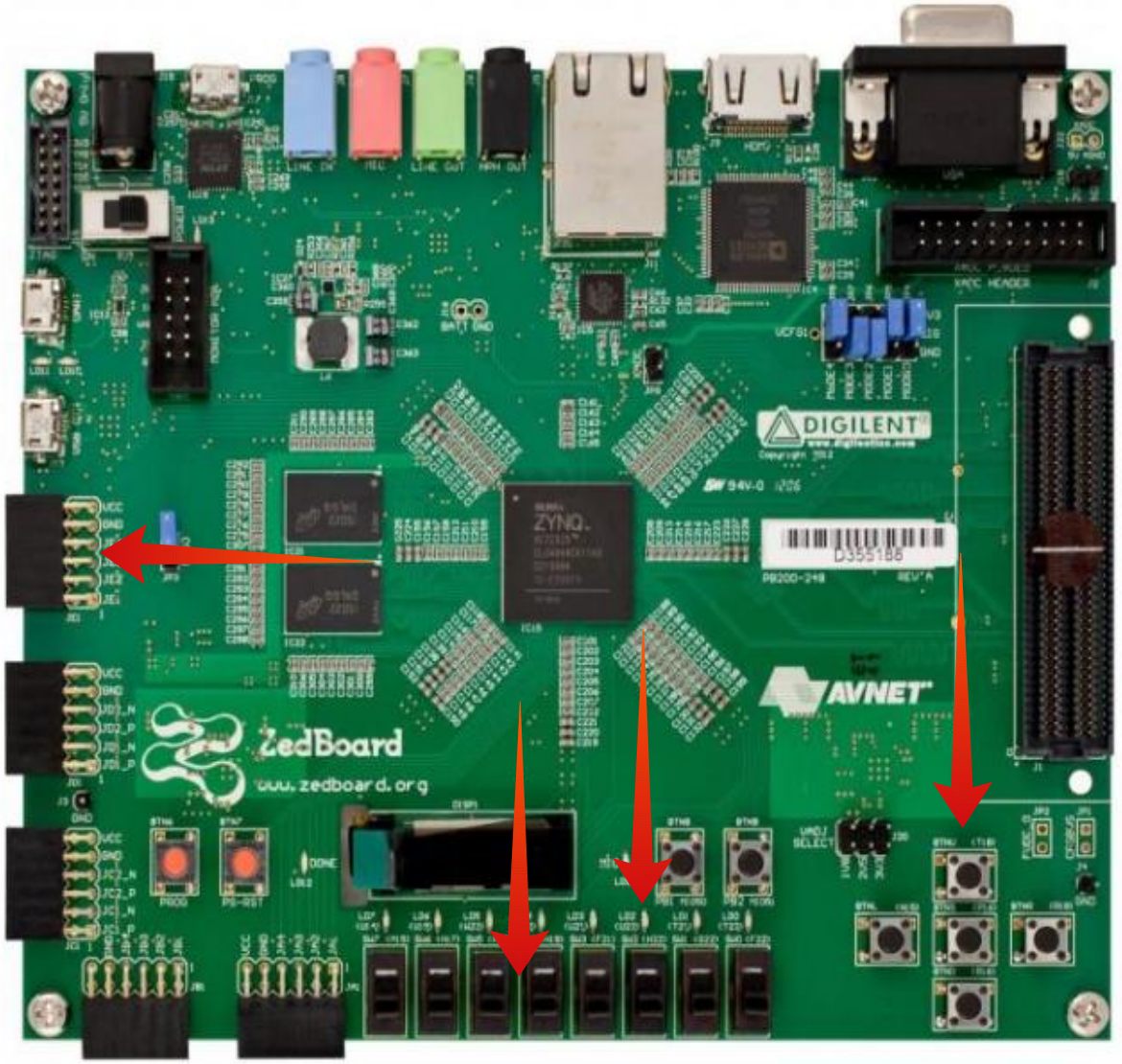


1. **USB-JTAG: permette di trasferire il Bitstream e di leggere il contenuto della DDR.**

2. **USB-UART: permette di ricevere e trasmettere dati in seriale.**

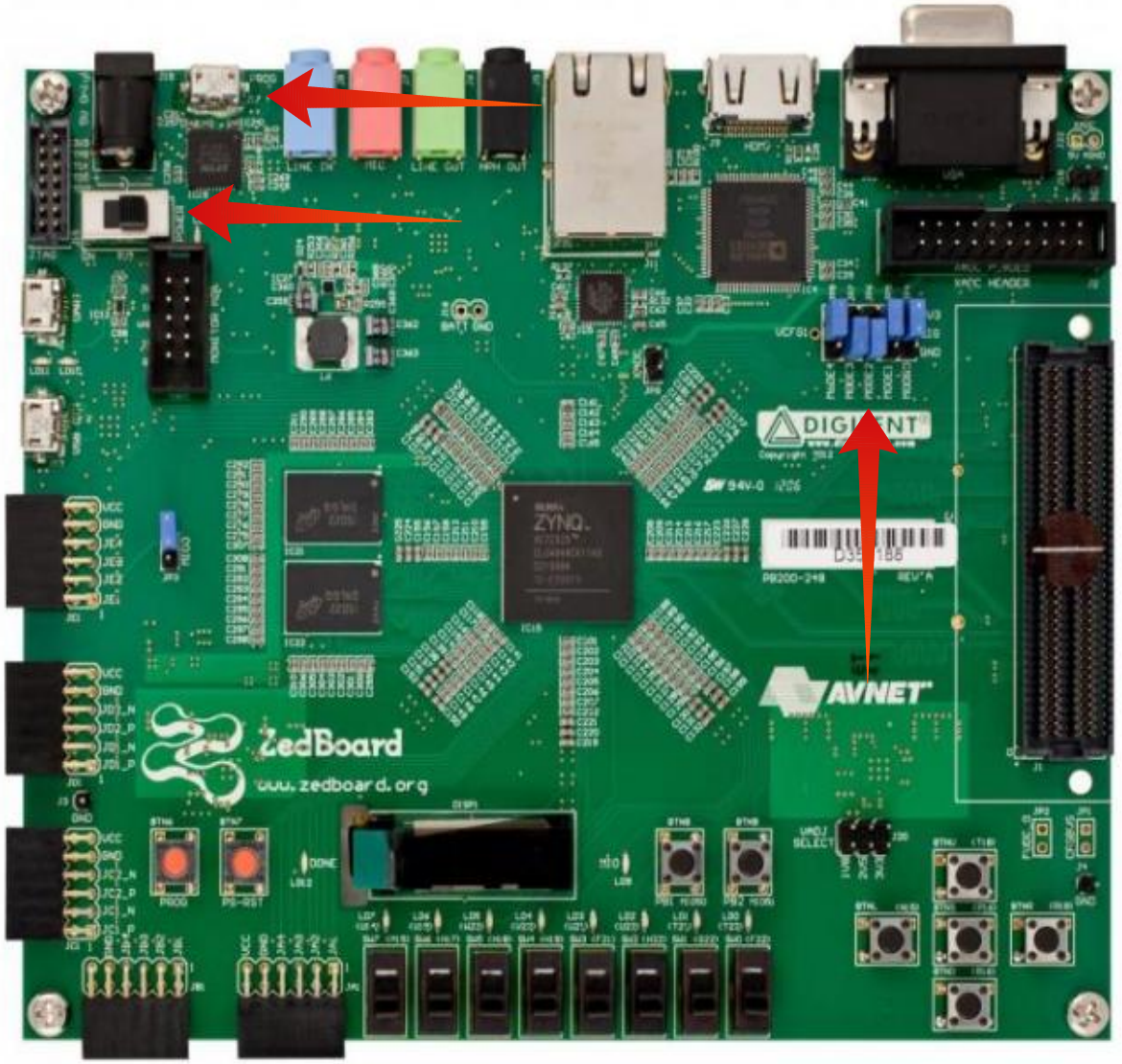
3. **USB-OTG: permette di collegare delle periferiche esterne.**

ZedBoard – IO Programmabili



1. **Pmod: ingressi e uscite programmabili classiche GPIO.**
2. **Switch: interruttori che programmabili come ingressi.**
3. **Pulsanti: altri input programmabili.**
4. **Led: possibile utilizzarli come uscite generiche.**

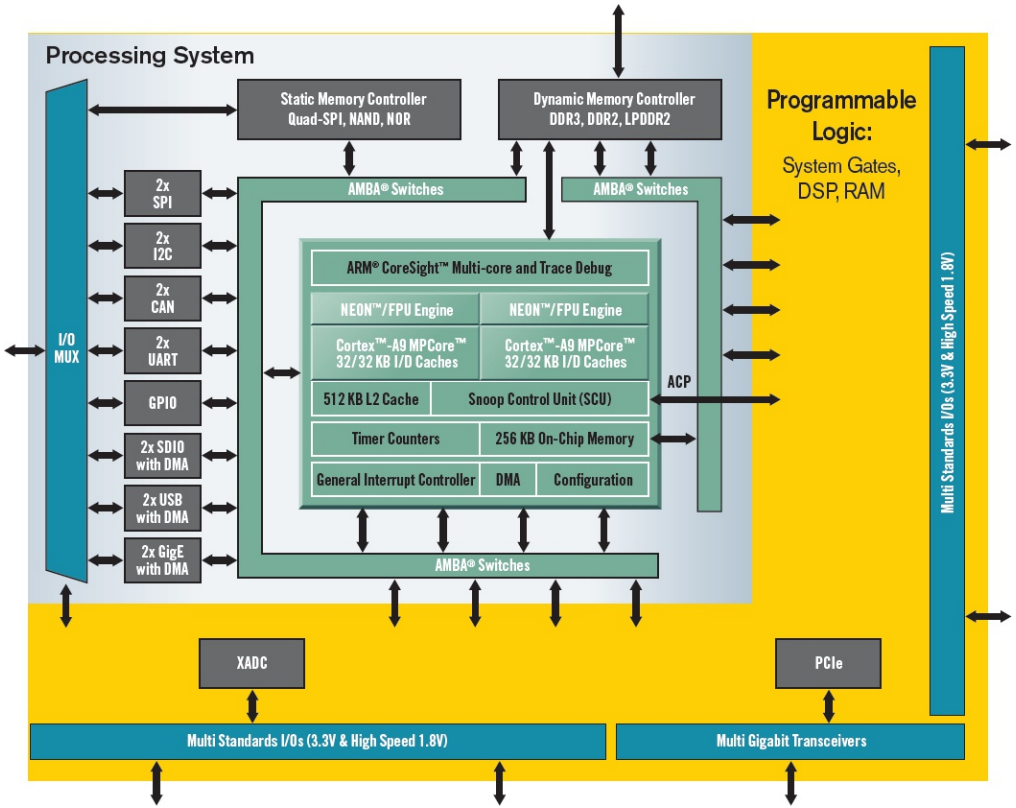
ZedBoard – Boot (JTAG)



1. **Collegare il cavo JTAG al proprio computer.**
2. **Portare i 5 jumper nella posizione bassa (come il secondo e il terzo).**
3. **Accendere la board.**

Hardware-Software Codesign

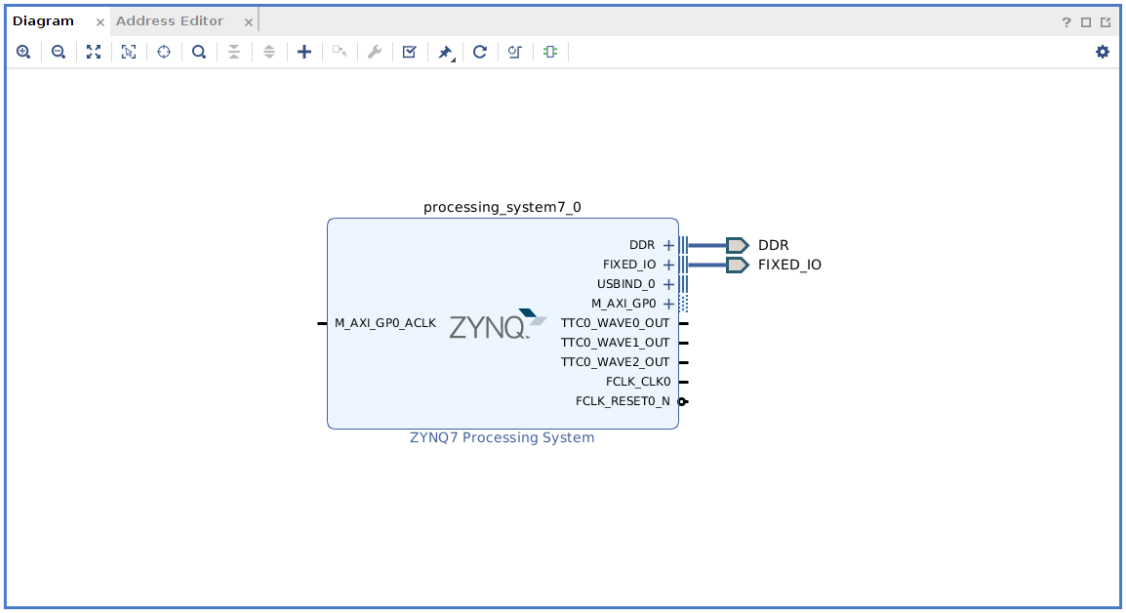
- Nei dispositivi FPGA-based moderni oltre alla logica programmabile sono presenti diverse entità: Cluster di CPU, DSPs, moduli ASIC ecc.



- **PL**: Programmable Logic;
- **PS**: Processing System.
- Queste due entità possono comunicare per mezzo di un protocollo definito da ARM: **AMBA AXI**.
- Disponibile in diverse versioni: **AXI Lite**, **AXI Full**, **AXI Stream**.

Configurazione PS

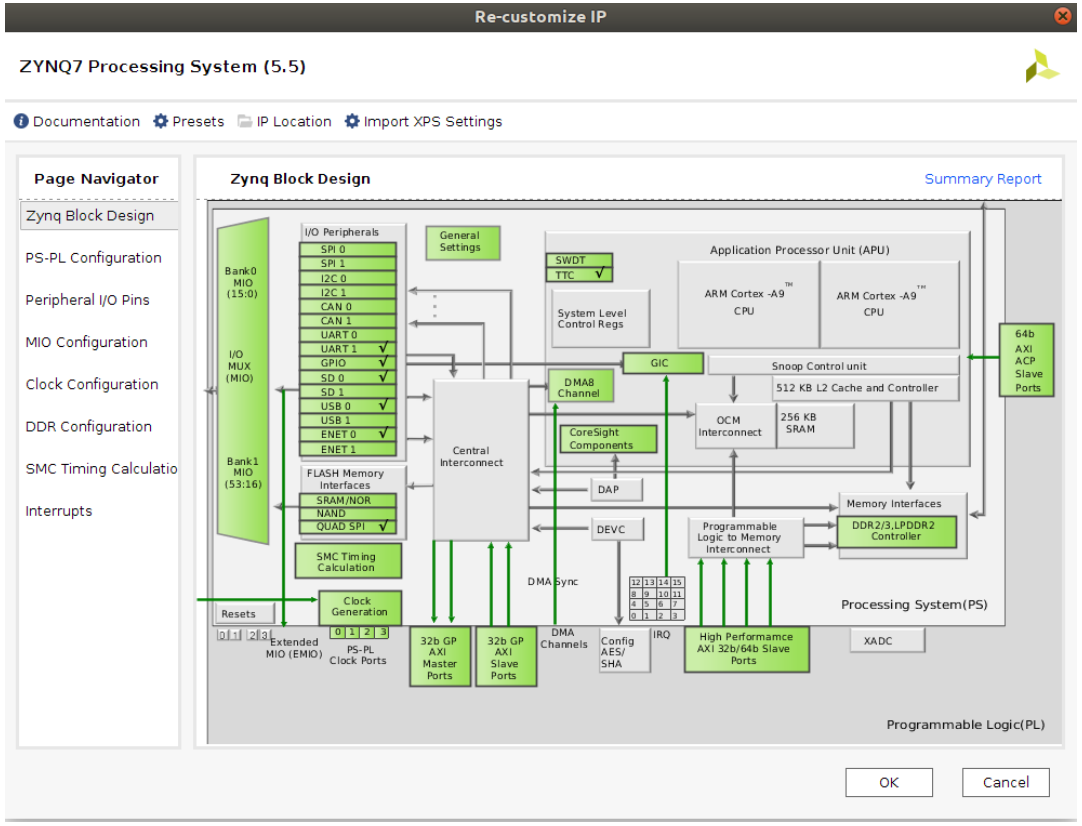
- Configurazione del Processing System tramite Vivado:
  - Aggiungere l'IP Zynq7 Processing System



- Clicchiamo su Run connect automation, in modo da collegarlo ad alcune periferiche necessarie: DDR e Fixed IO.

Configurazione PS

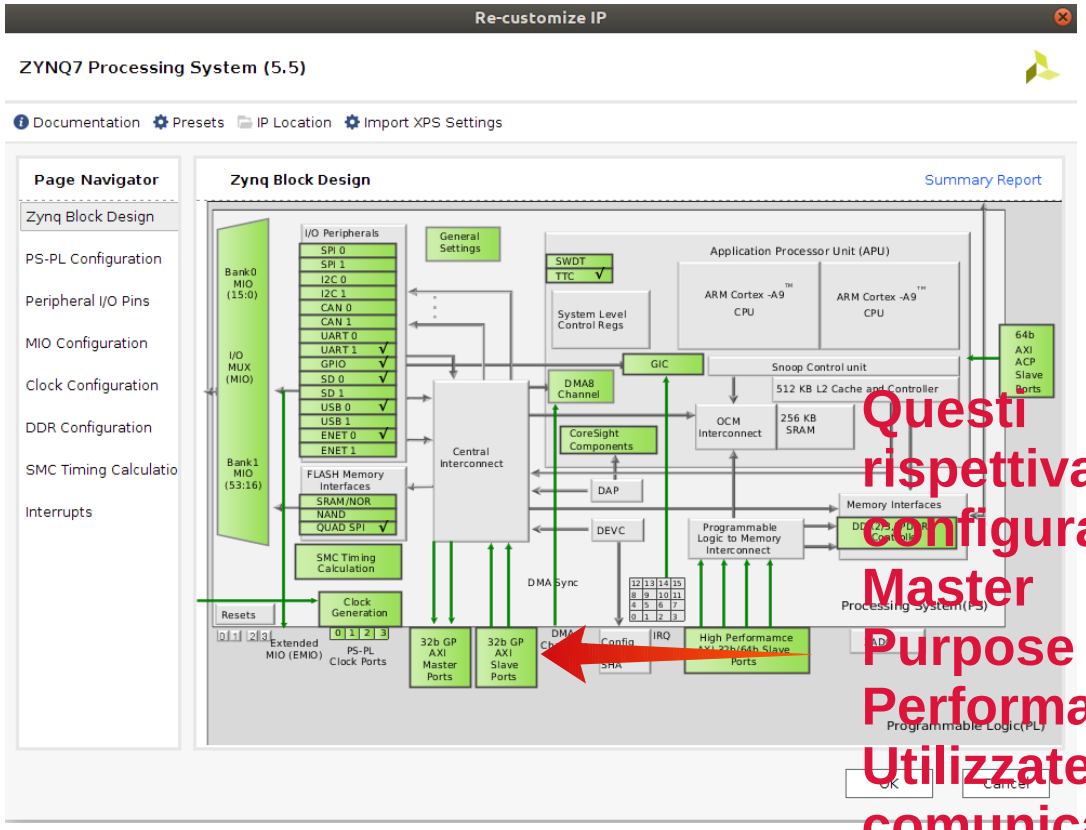
- Doppio Click sul modulo, per entrare nelle impostazioni:



- Lo schema in figura è lo schema a blocchi del PS.
- Tutto ciò che è riportato in verde è configurabile.
- Vediamo solamente le cose fondamentali.

Configurazione PS – GP/HP Ports

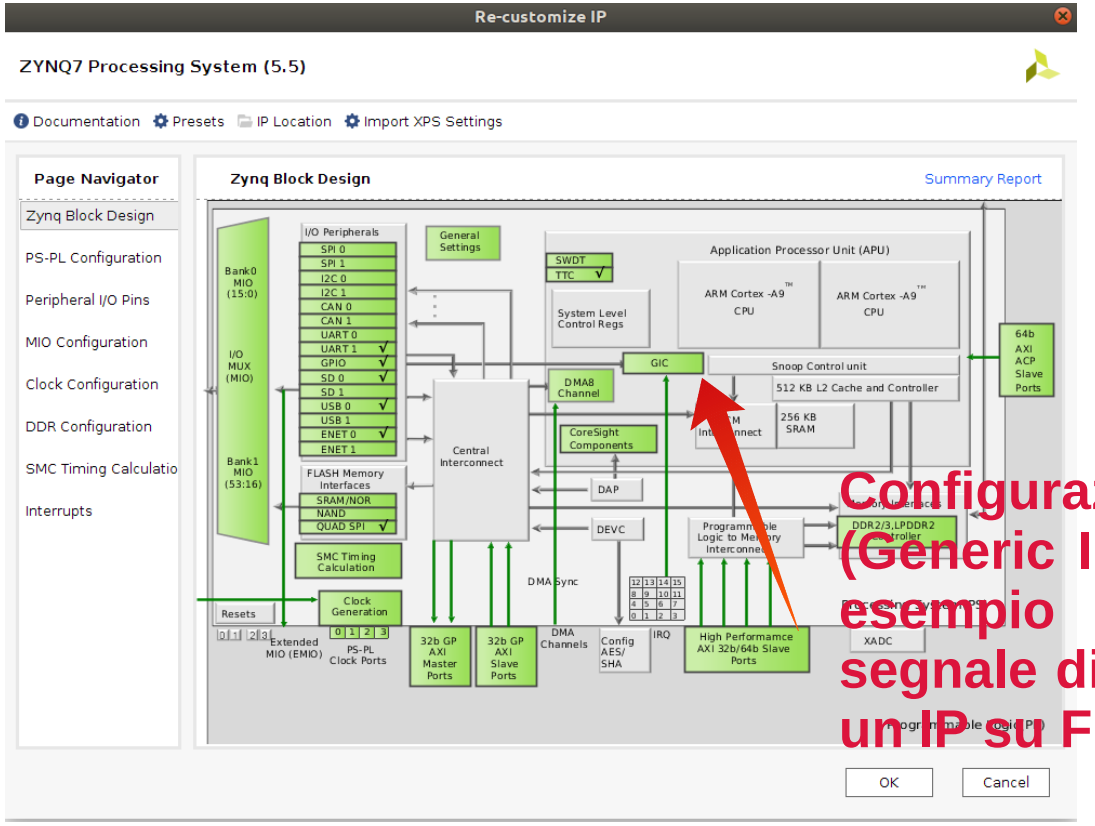
- Doppio Click sul modulo, per entrare nelle impostazioni:



Questi due blocchi rispettivamente sono per la configurazione di porte Master / Slave General Purpose (GP Ports) e High Performance (HP Ports). Utilizzate da AXI per comunicare con la PL.

Configurazione PS - GIC

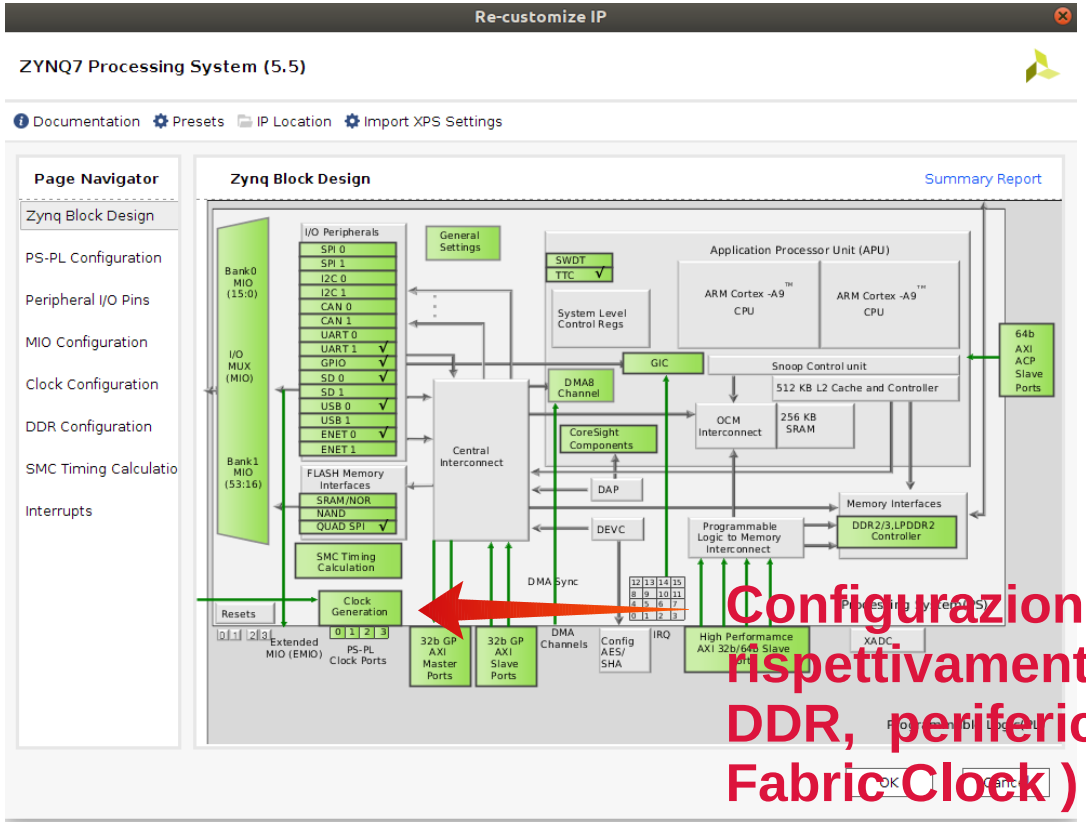
- Doppio Click sul modulo, per entrare nelle impostazioni:



Configurazione del GIC (Generic Interrupt Controller), esempio per ricevere un segnale di termine da parte di un IP su FPGA.

Configurazione PS – Clock

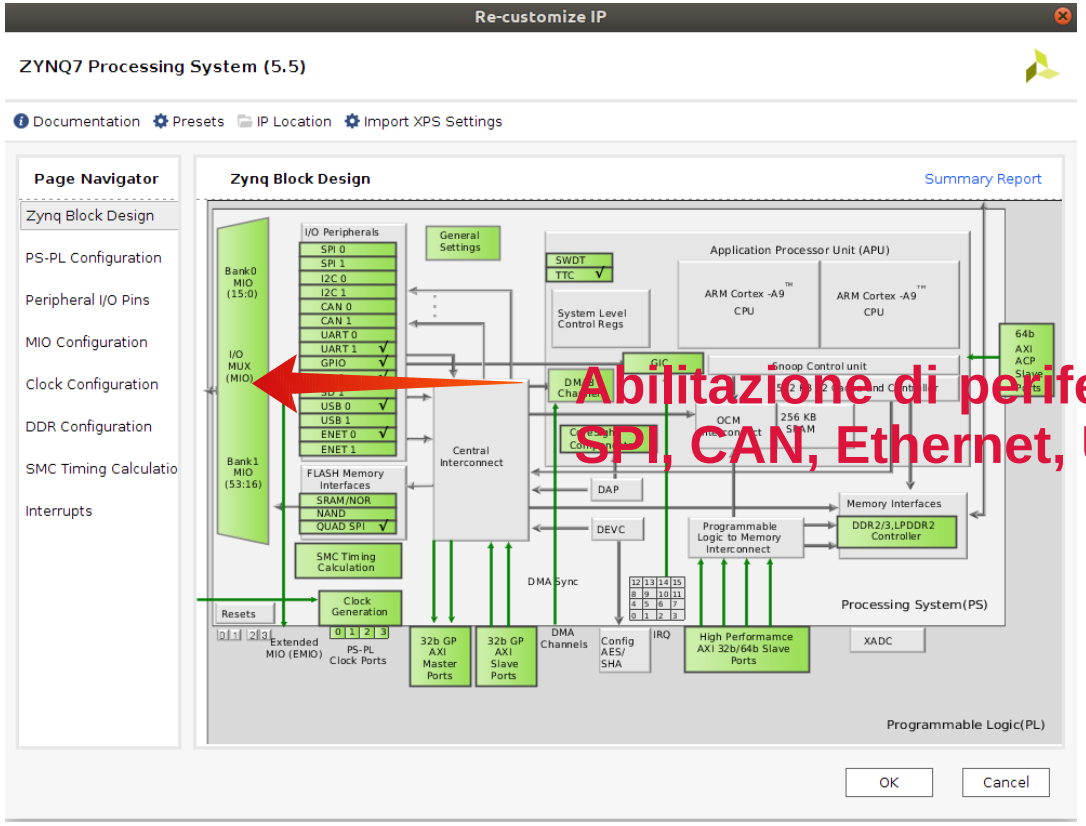
- Doppio Click sul modulo, per entrare nelle impostazioni:



Configurazione del Clock, rispettivamente per Cores ARM, DDR, periferiche e FPGA (PL Fabric Clock)

Configurazione PS – Periferiche

- Doppio Click sul modulo, per entrare nelle impostazioni:



Abilitazione di periferiche, esempio: I2C, SPI, CAN, Ethernet, USB, ...

Comunicazione PS-PL: AMBA AXI

- interfacce di comunicazione, definite da ARM, e ora standard per comunicazioni on-chip.

- **AXI4**: Fornisce elevate prestazioni per accessi di tipo **memory mapped**. Permette di eseguire fino a 256 trasferimenti con singolo indirizzamento (burst) e altre caratteristiche avanzate per ottenere **alte prestazioni**.
- **AXI4-Lite**: Consente di eseguire singoli (no burst) trasferimenti di tipo **memory mapped** che richiedono **basso livello di throughput**. Spesso utilizzato per la scrittura e la lettura di registri di stato e di controllo.
- **AXI4-Stream**: Per streaming ad **alte prestazioni**. **Non prevede indirizzamento** (i.e. non è di tipo memory mapped). Consente trasferimenti dati di dimensione illimitata. Utilizzato tipicamente, ma non solo, per trasferire flussi di immagini.

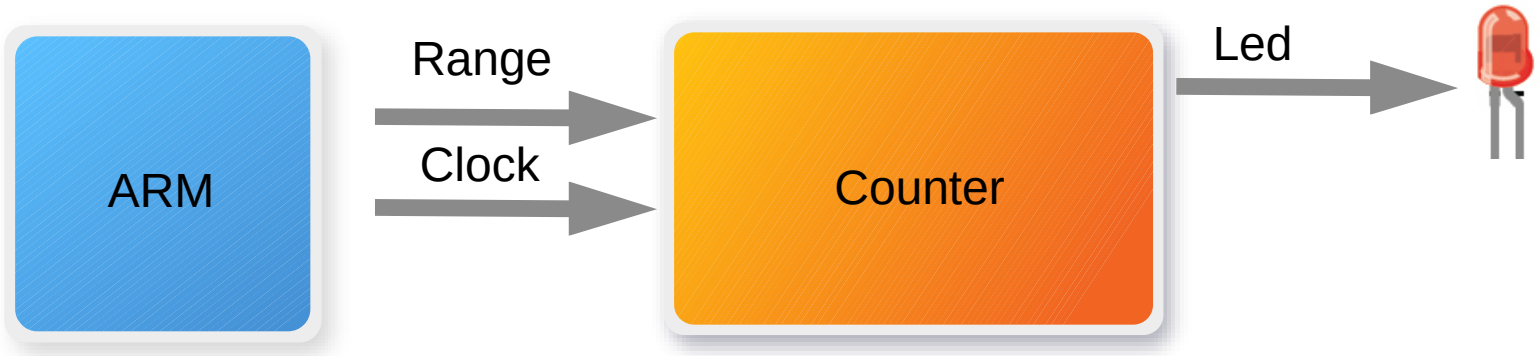
Comunicazione PS-PL: AMBA AXI

- Le comunicazioni AXI4 sono di tipo **Master-Slave** bidirezionali, nel senso che la comunicazione è iniziata da una periferica master.
- Per utilizzare AXI4 tramite HLS, è necessario marcare una particolare porta di ingresso tramite le seguenti pragma:

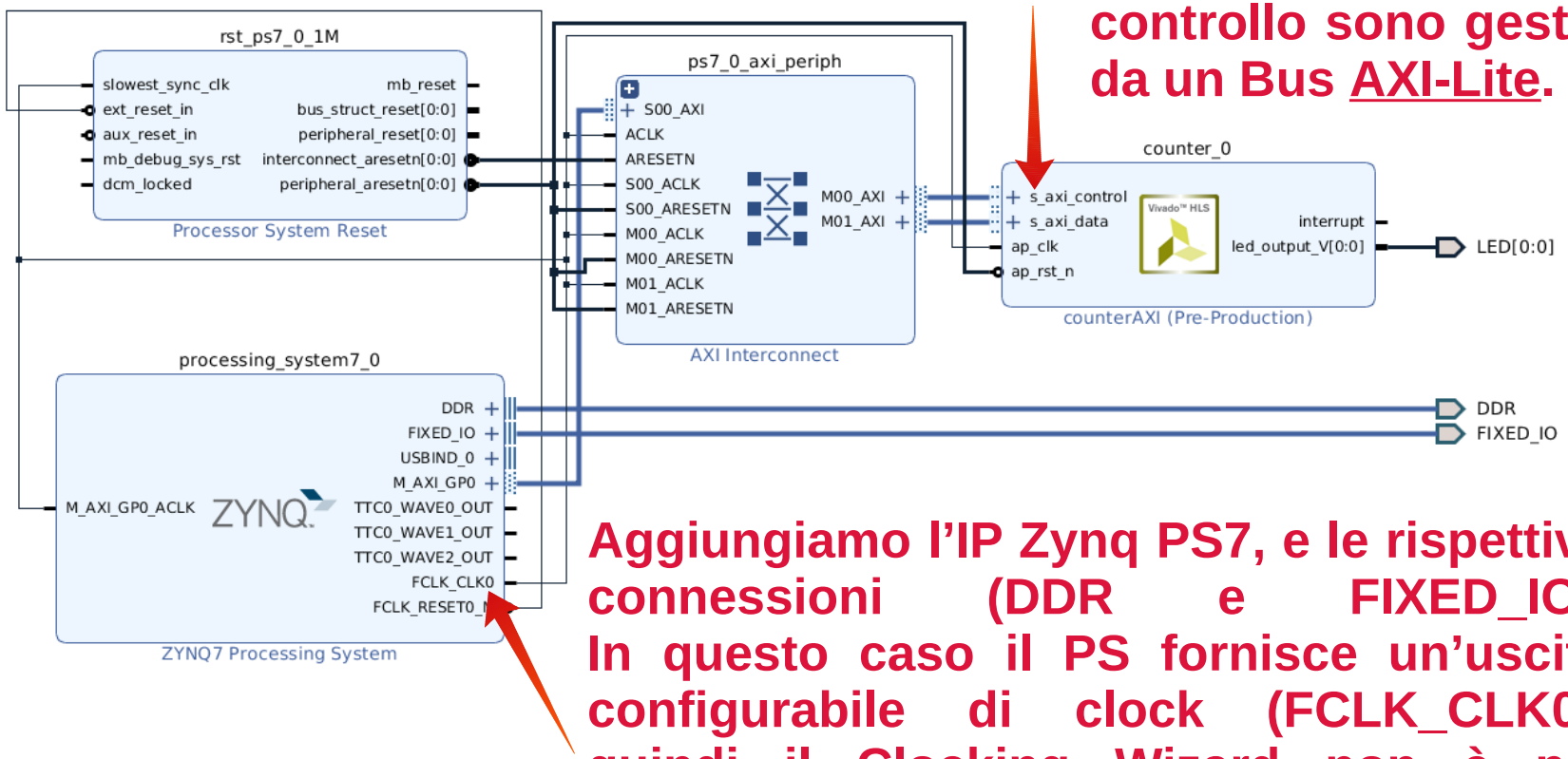
```
1   #pragma HLS INTERFACE s_axilite port=myport bundle=mybundle
2   #pragma HLS INTERFACE m_axi     port=myport bundle=mybundle
3   #pragma HLS INTERFACE axis      port=myport bundle=mybundle
```

HLS – Esercitazione 02

- Provare a realizzare in FPGA un modulo contatore che dopo N cicli di clock vada ad invertire lo stato di un led, mappato sulla ZedBoard.
- Estendiamo la precedente esercitazione in modo tale che sia il PS a passare il range al modulo FPGA, attraverso AXI-Lite:



Creazione Block Design - Differenze



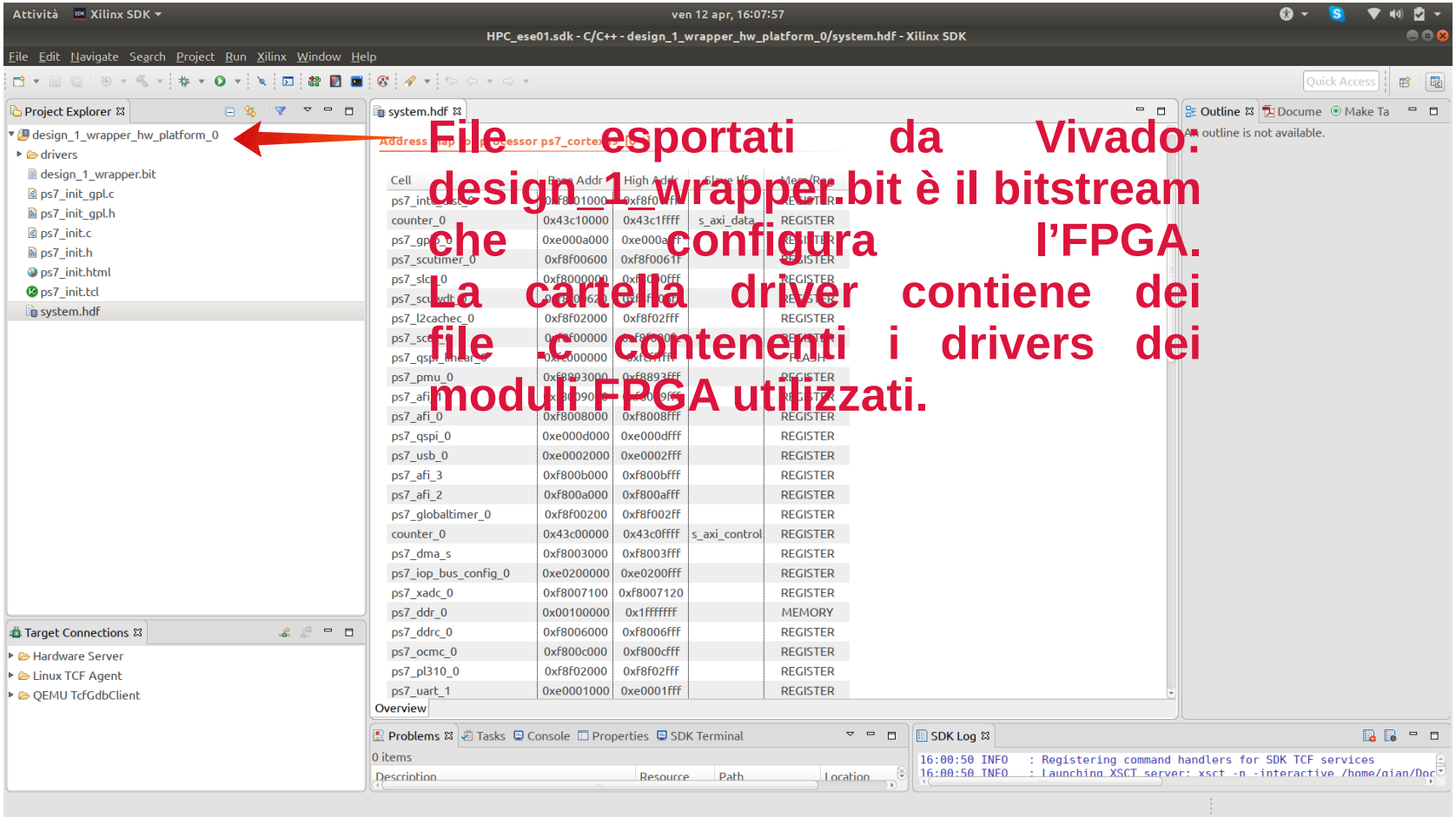
Tutti i segnali di controllo sono gestiti da un Bus AXI-Lite.

Aggiungiamo l'IP Zynq PS7, e le rispettive connessioni (DDR e FIXED_IO). In questo caso il PS fornisce un'uscita configurabile di clock (FCLK_CLK0), quindi il Clocking Wizard non è più necessario.

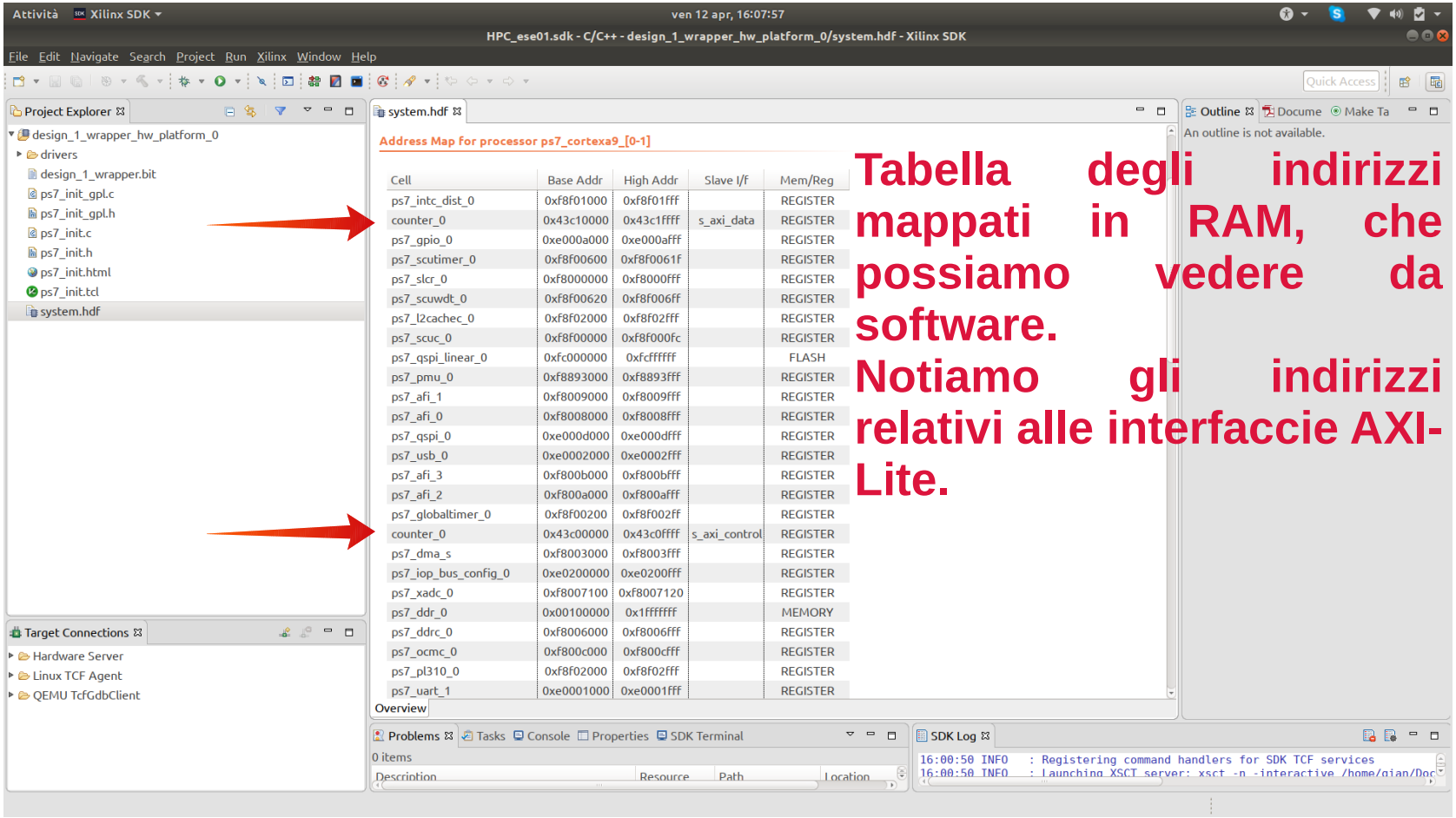
Esportiamo l’Hardware

- Dopo aver generato il bitstream, invece che programmare direttamente il dispositivo andiamo ad esportare l’hardware generato:
- File > Export > Export Hardware.
- File > Launch SDK
- In questo modo verrà lanciata un’istanza del software **Xilinx SDK** avente come workspace la directory del progetto Vivado che abbiamo creato.
- Tramite XSDK andiamo a creare il software per ARM (Baremetal) che ci servirà per controllare il modulo FPGA.

XSDK Overview - BSP

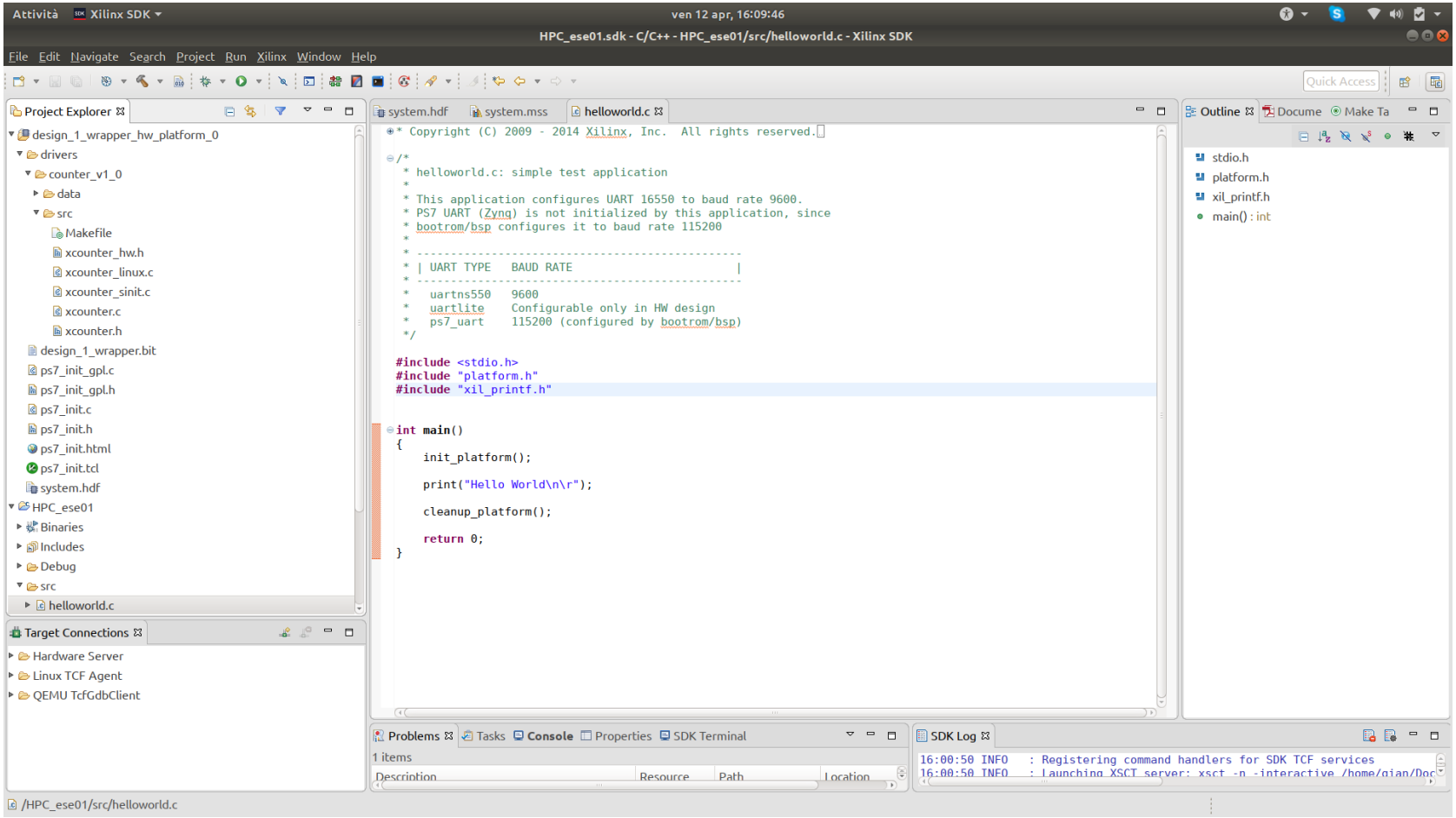


XSDK Overview - BSP



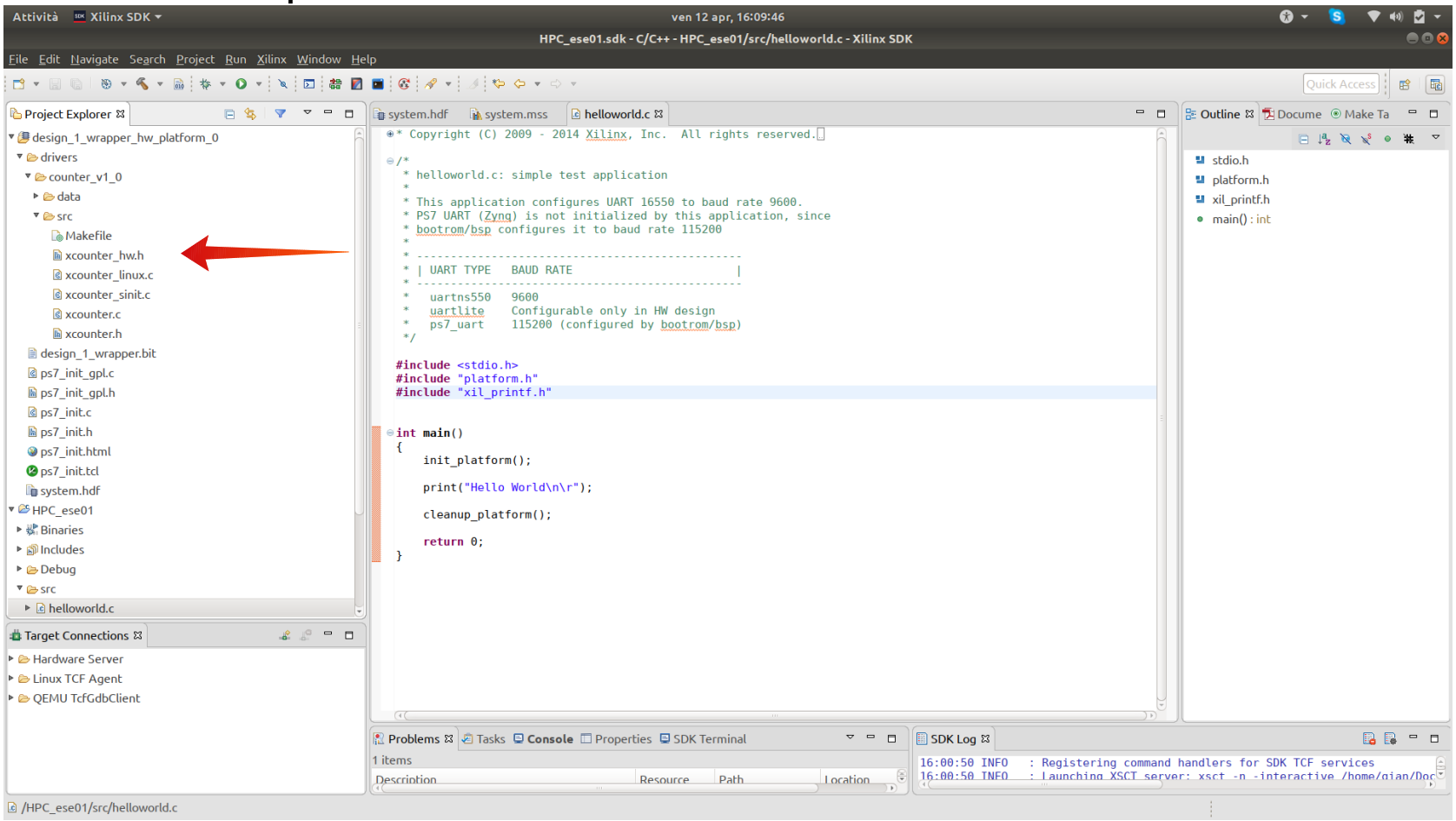
XSDK Overview - Progetto

- Creiamo un nuovo “Application Project”, avente come template “Hello World”



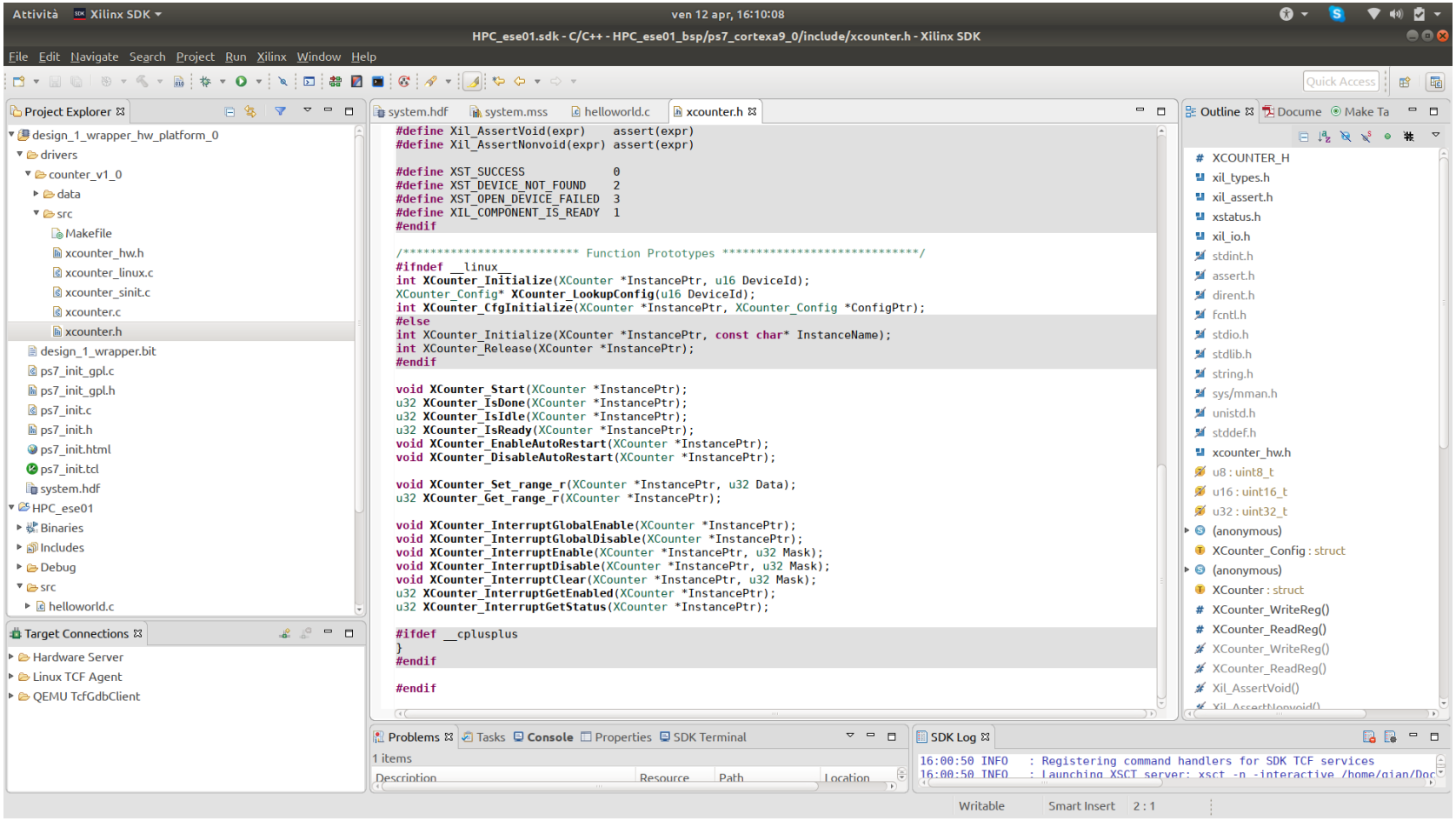
XSDK Overview - Drivers

- Notiamo i file include all'interno della cartella "Drivers", dovremo includerli per usare il modulo contatore.



XSDK Overview - Drivers

- Drivers principali:



XSDK Overview - Drivers

- I driver solitamente sono nominati x<nome-modulo>.h /.c
- Nel nostro caso i principali sono **xcounter** e **xcounter_linux**, che sono rispettivamente utilizzati in caso di Baremetal e Linux.
- Nel caso di Linux, gestiscono la traduzione da indirizzo fisico a indirizzo virtuale, ovviamente in caso di progetto Linux, dovremo anche inserire questi indirizzi nel Device Tree del dispositivo.
- Per semplicità noi lavoreremo in Baremetal.

XSDK Overview – Codice ARM

- Realizziamo quindi il software di controllo come da questo esempio:

```c
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "xcounter.h"

int main() {

    init_platform();

    XCounter counterInstance;

    u32 range = 100000;

    XCounter_Initialize(&counterInstance, 0);

    printf("Counter initialized, range: %d\n\r", range);

    XCounter_Set_range_V(&counterInstance, range);

    printf("%x\n", Xil_In32(0x0));

    while(1) {

        XCounter_Start(&counterInstance);

        while(!XCounter_IsDone(&counterInstance));
        printf("Counter Done!\n\r");
    }
    cleanup_platform();

    return 0;
}
```

HLS – Esercitazione 02.5

- Nella precedente esercitazione il controllo del modulo FPGA è gestito secondo una logica di **Polling**, ovvero il processore deve svolgere dei cicli nel quale va a controllare lo stato della perfierica.
- Provare a modificare il design dell'esercitazione precedente, in modo tale da sfruttare l'uscita interrupt del modulo FPGA e il Generic Interrupt Controller della PS.
- Un esempio di configurazione del GIC è possibile trovarlo sul sito del corso.