UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Dipartimento di Scienze Fisiche,
Informatiche e Matematiche

6. HLS Lab #2

High Performance Computing [262-022]

Dott. Gianluca Brilli
gianluca.brilli@unimore.it
Dott. Alessandro Capotondi
Alessandro.capotondi@unimore.it
Prof. Andrea Marongiu
andrea.marongiu@unimore.it

Corso di Laurea in INFORMATICA
(D.M.270/04) [16-262]
Anno accademico 2020/2021

Agenda

Cosa vedremo in queste lezioni

- ➔ *Esempi di accelerazione tramite FPGA:*
  - ➔ *Simulazione di alcuni kernel computazionali, applicando alcune ottimizzazioni HLS.*

- ➔ ***Test su development board:***
  - ➔ ***Realizzazione di un design completo e test su piattaforma di sviluppo.***

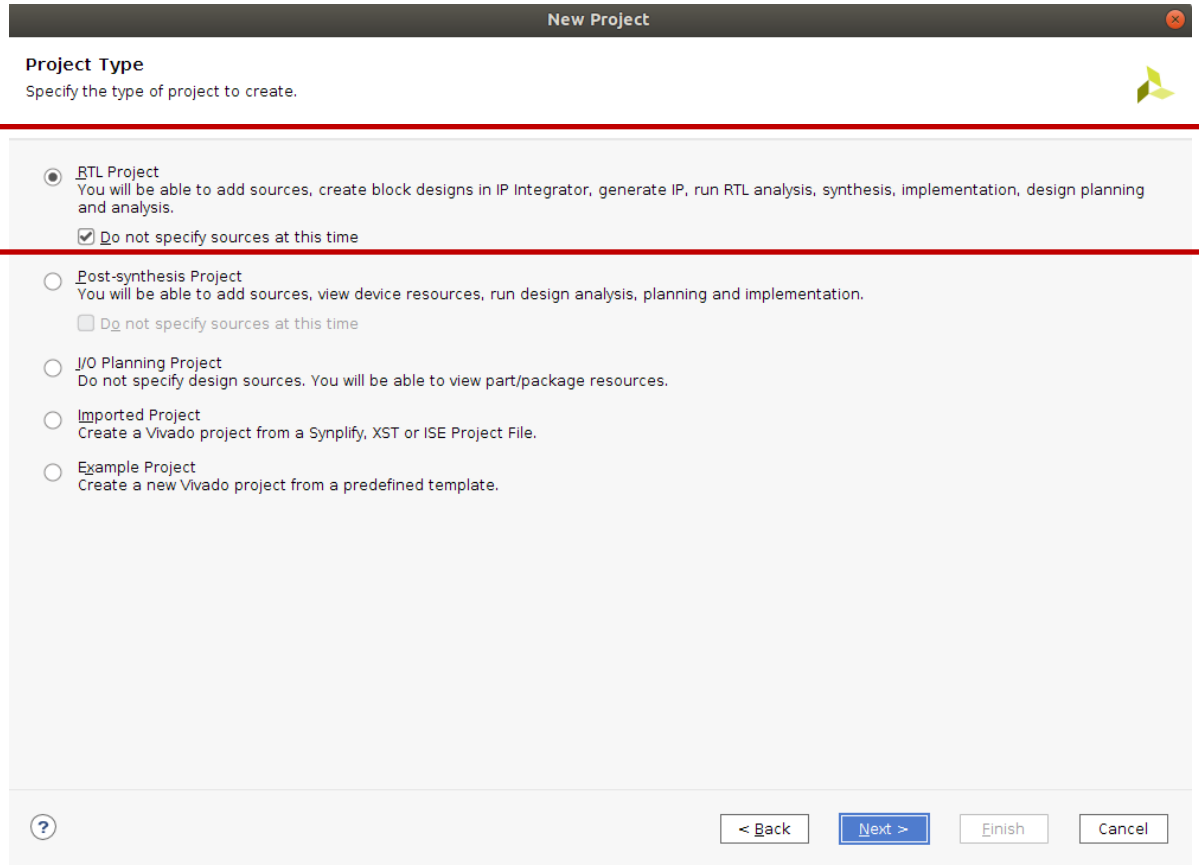
Esempio

- *Realizzazione di un progetto che ci permetta di eseguire del codice su development board.*
- *Scrittura sul terminale della stringa «Hello World».*

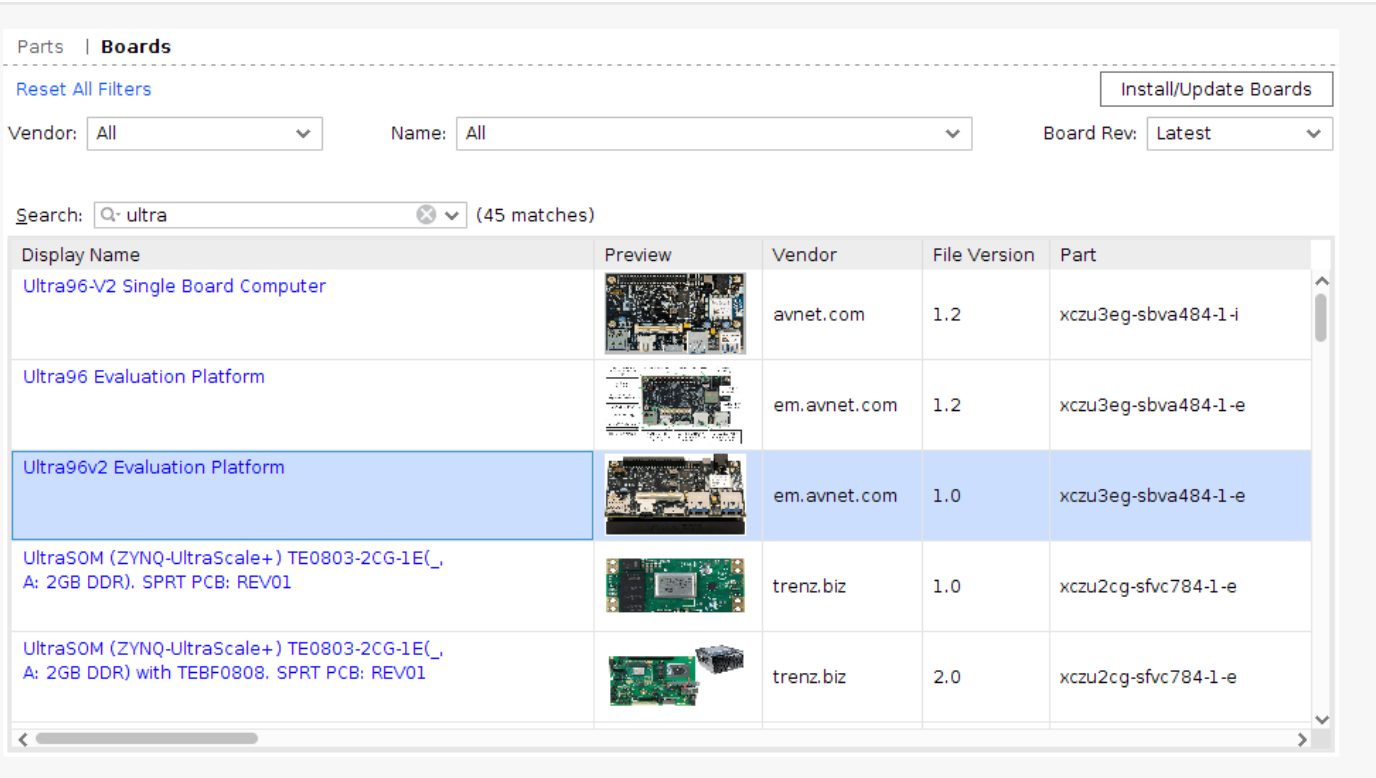
Esempio

Creazione di un progetto Vivado

- ➔ *$ source /tools/Xilinx/Vivado/2020.1/settings64.sh*
- ➔ *$ vivado*



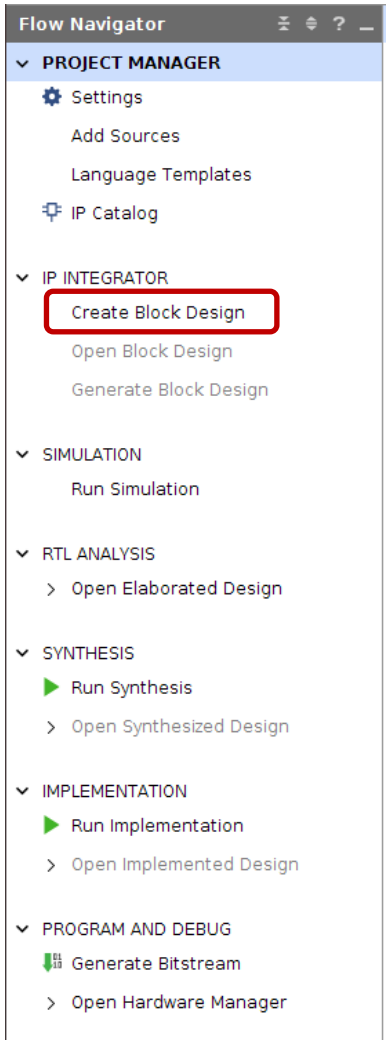
Esempio
Creazione progetto



- *Selezionare Ultra96v2 Evaluation Platform;*
- *Next e Finish.*

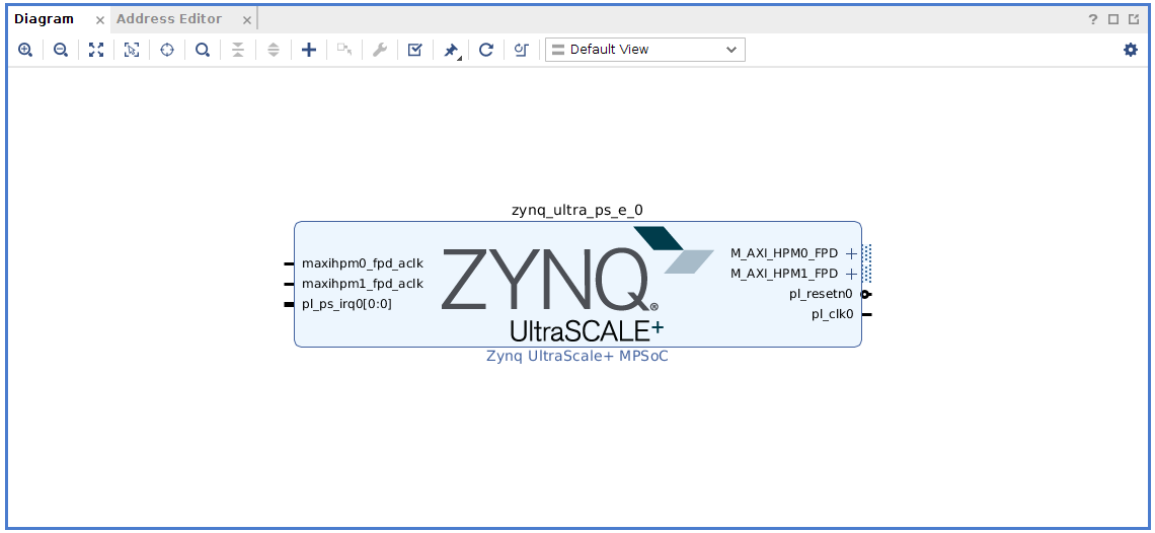
Esempio

Creazione block design



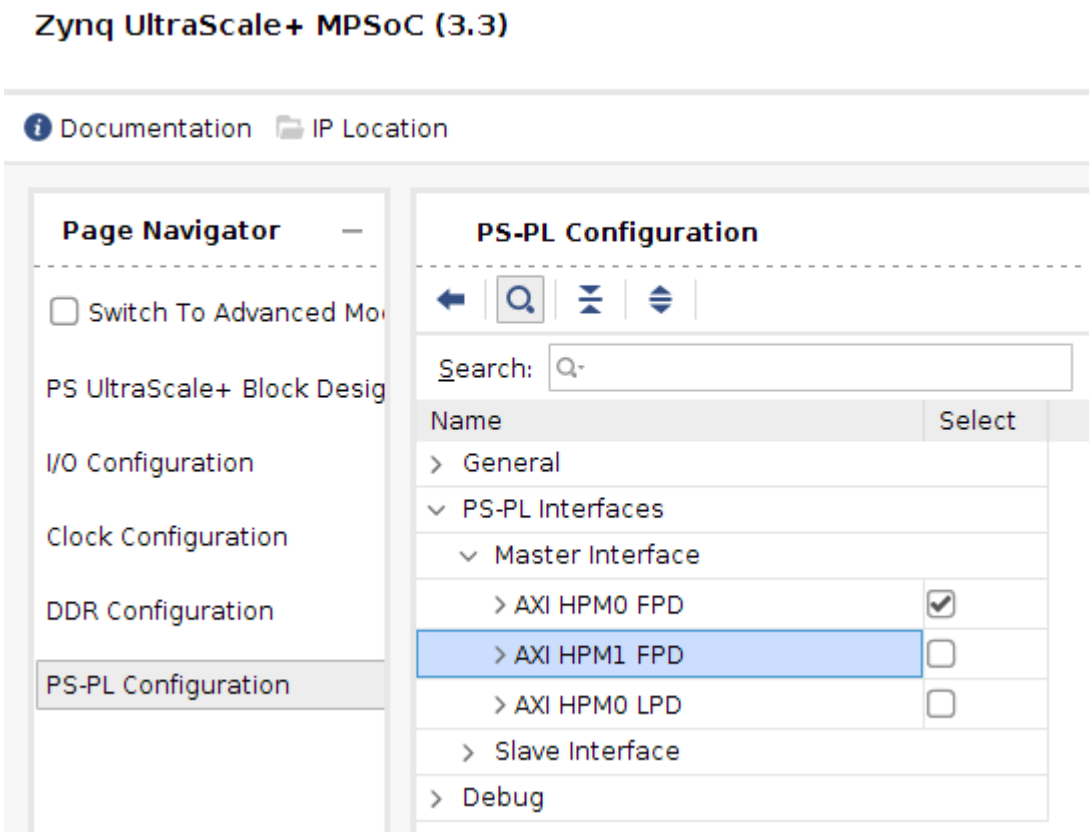
1. *Cliccare su «Create Block Design»;*
2. *Sul design creato cliccare sul «+» (Add IP);*
3. *Aggiungere l'IP «Zynq UltraScale+ MPSoC»;*
4. *Cliccare su «Run Block Automation»;*
5. *Apply Board Preset;*
6. *Ok.*

Esempio
Creazione block design



➔ *Customizziamo l'IP, rimuovendo tutto ciò che non serve al momento.*

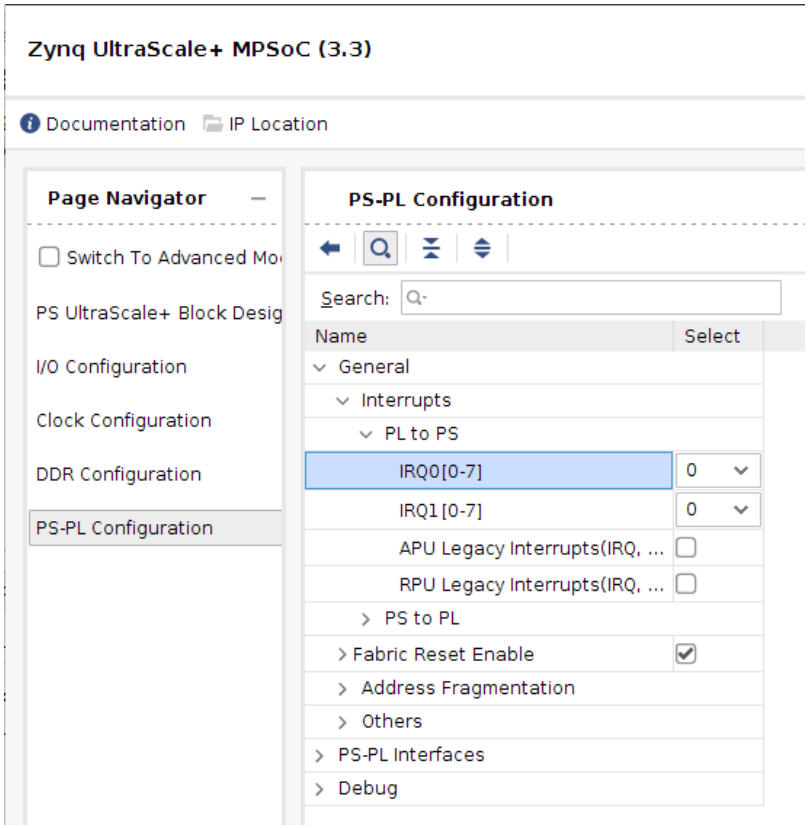
Esempio
Rimozione di una porta master



- ➔ *Doppio click sull'IP per aprire la schermata di configurazione;*
- ➔ *PS-PL Configuration;*
- ➔ *Disabilitare «AXI HPM1 FPD»*

- ➔ *Le AXI HPM sono utilizzate per la configurazione di registri dei moduli FPGA.*
- ➔ *Al momento una è sufficiente.*

Esempio
Rimozione dell'interrupt

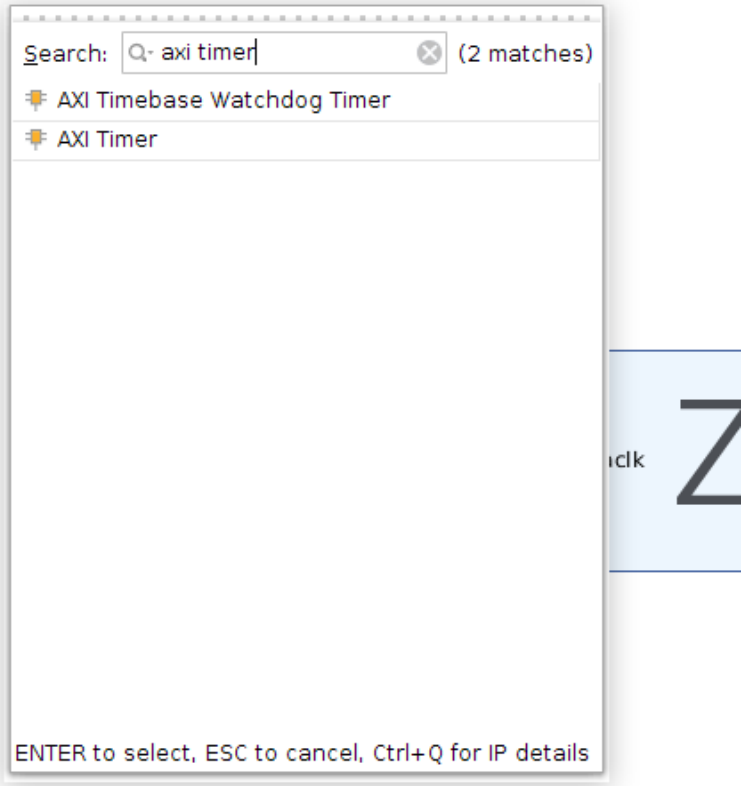


- ➔ *Doppio click sull'IP per aprire la schermata di configurazione;*
- ➔ *PS-PL Configuration;*
- ➔ *Disabilitare «IRQ 0 [0-7]»*

➔ *Interrupt non utilizzato al momento.*

Esempio

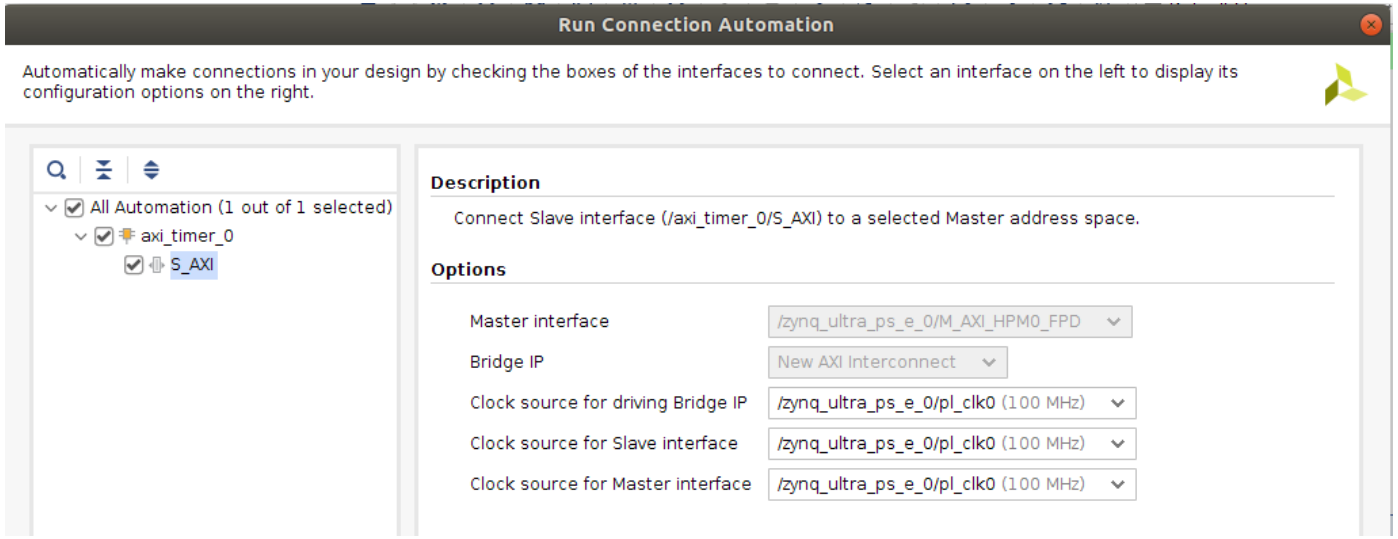
Timer per profiling



- ➔ *Aggiungiamo anche l'IP AXI Timer;*
- ➔ *Ci servirà per profilare i tempi di esecuzione.*

Esempio

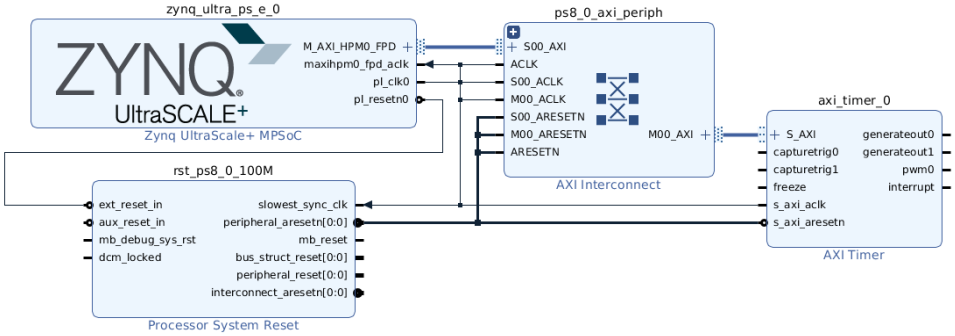
Timer per profiling



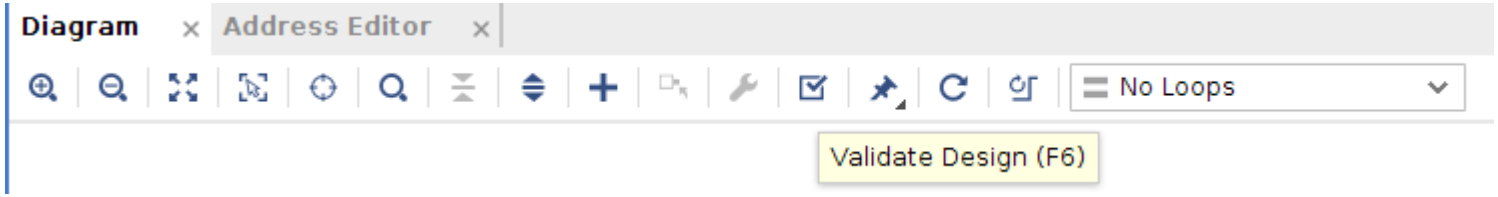
➔ *Cliccare su «Run Connection Automation» ed impostare le seguenti opzioni.*

➔ *A questo punto il design è completo.*

Esempio
Design finale e validazione

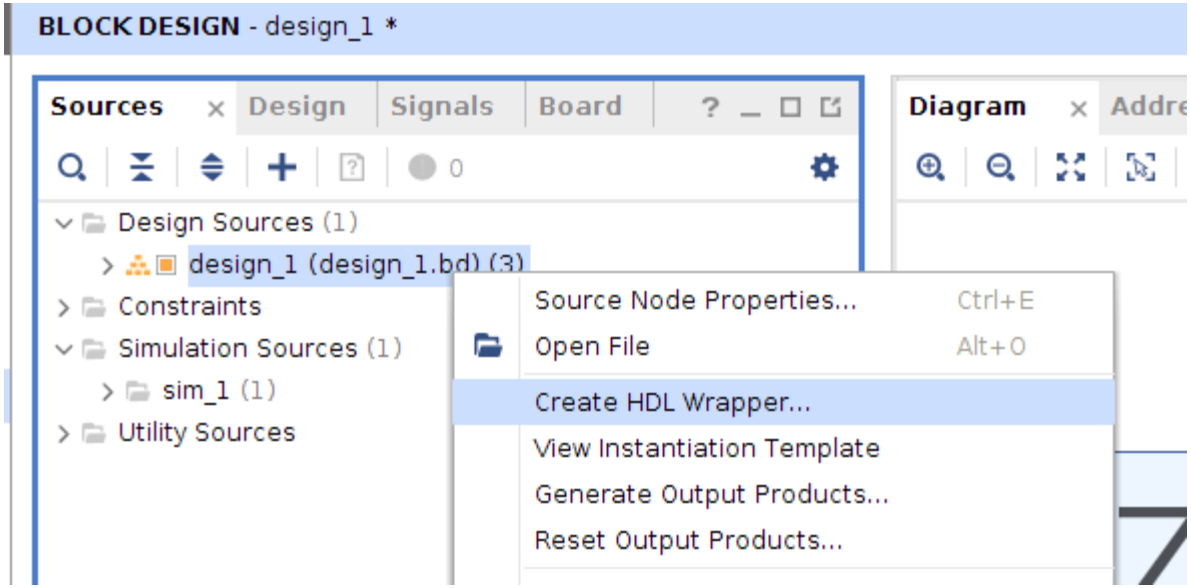


➔ *Cliccare su «validate design».*



Esempio
HDL wrapper

- ➔ *Se la validazione ha avuto esito positivo, procedere creando l'HDL Wrapper.*
- ➔ *Un file Verilog che va ad istanziare i blocchi posizionati all'interno del design.*

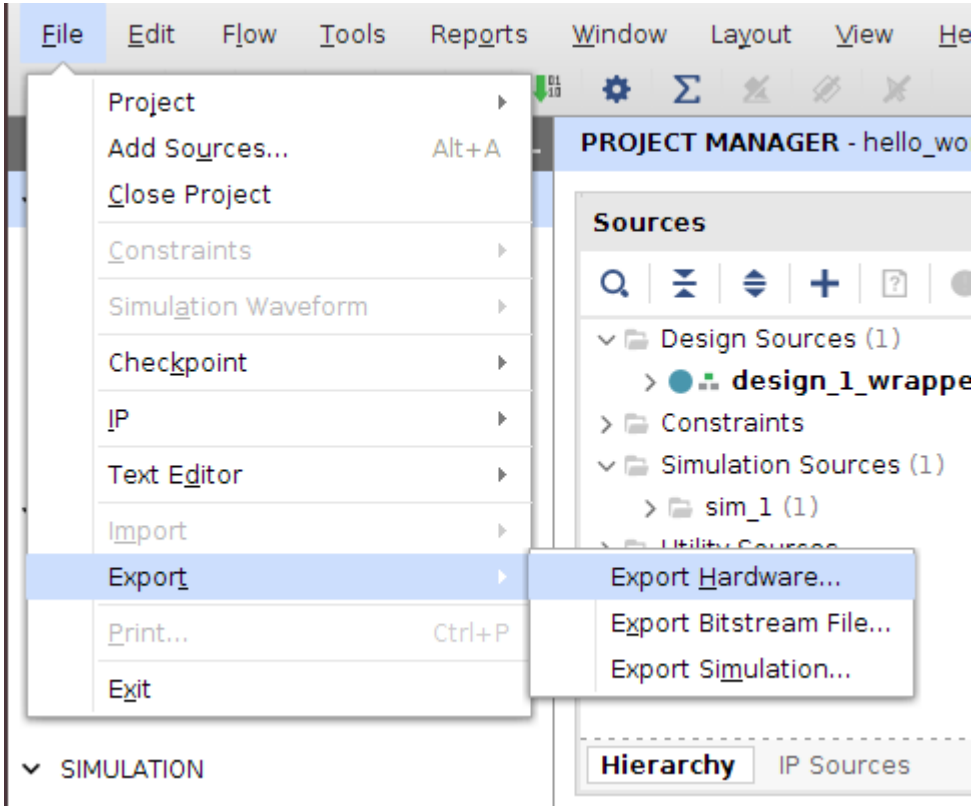


Esempio
Sintesi

- ➔ *Infine cliccare su «Program and Debug» > «Generate Bitstream»;*
- ➔ *Questa fase si compone di più sottofasi:*
  - ➔ *Block Design generation: i singoli blocchi vengono sintetizzati in parallelo;*
  - ➔ *Sintesi: sintesi globale del sistema;*
  - ➔ *Implementazione: il design sintetizzato viene infine implementato sulla piattaforma target del progetto.*

Esempio
Export hardware

➔ *Una volta terminato il processo di sintesi esportare l'hardware:*

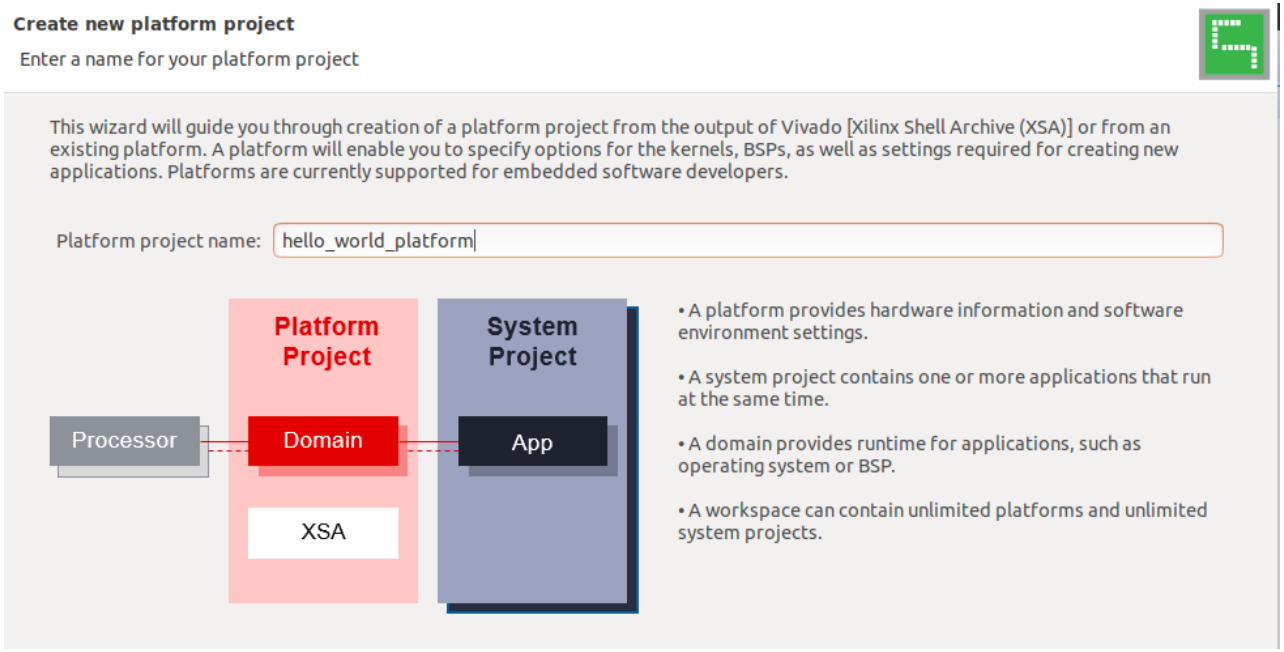


➔ *Come opzioni di export selezionare:*

- ➔ *Fixed;*
- ➔ *Include Bitstream.*

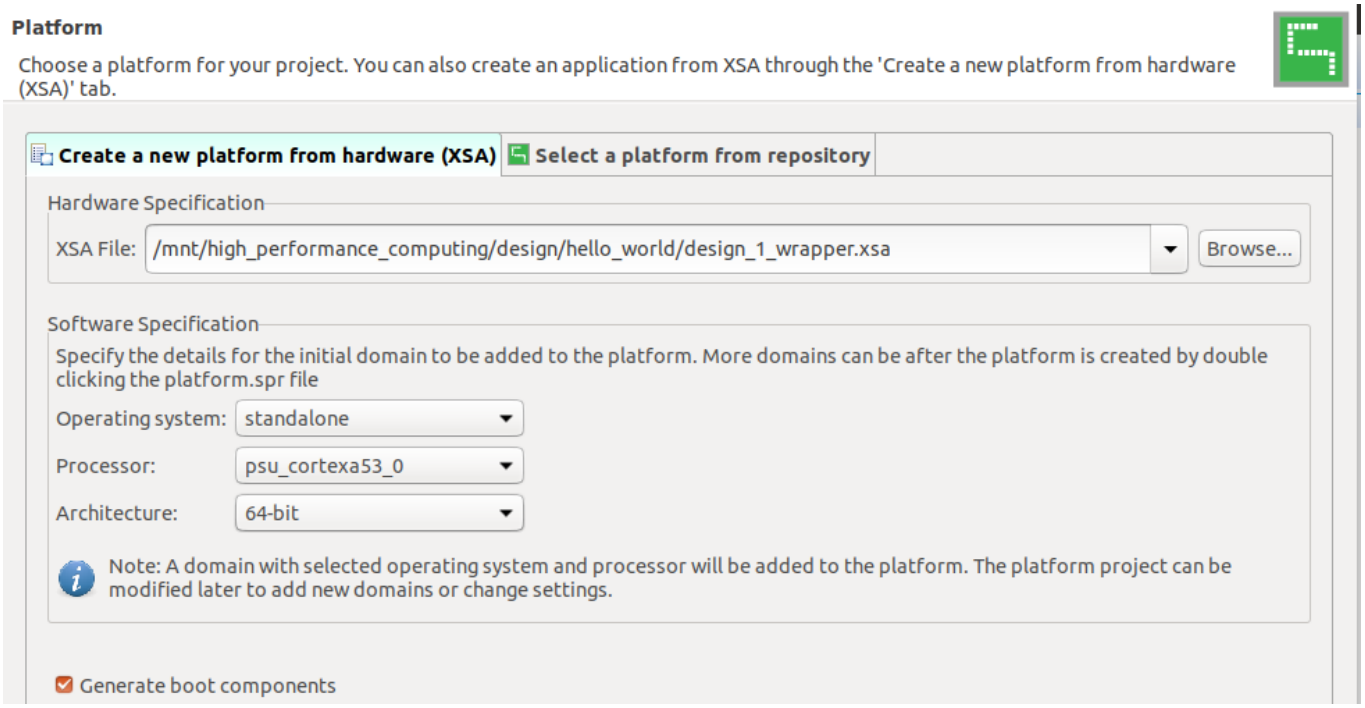
Esempio
Importazione in Vitis

➔ *Avviare Vitis SDK, cliccando su: Tools > Launch Vitis IDE*
➔ *Cliccare su New > Create Platform Project*



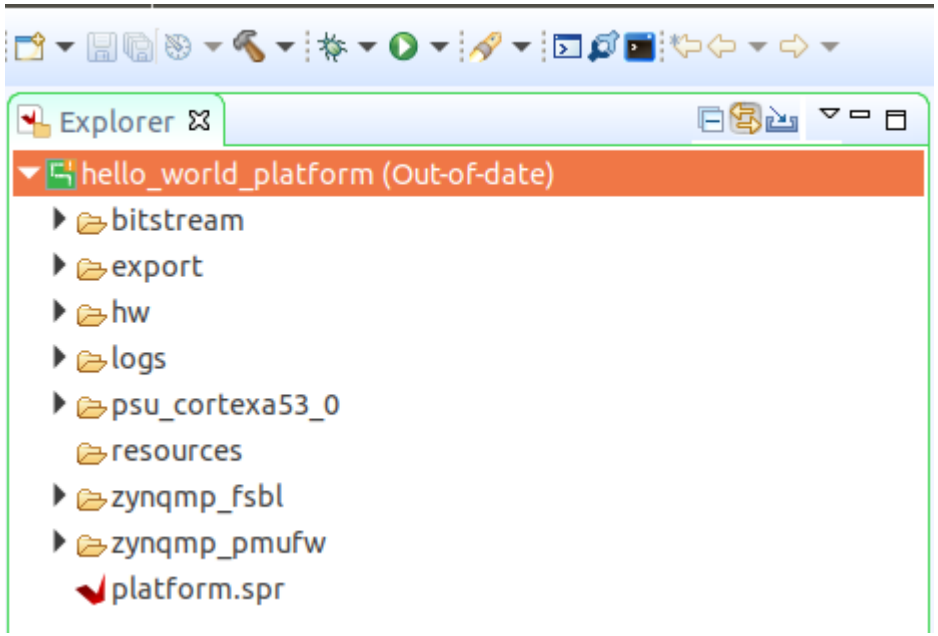
Esempio
Importazione in Vitis

➔ *Importare il file .xsa esportato da Vivado ed impostare i seguenti parametri:*



Esempio
Importazione in Vitis

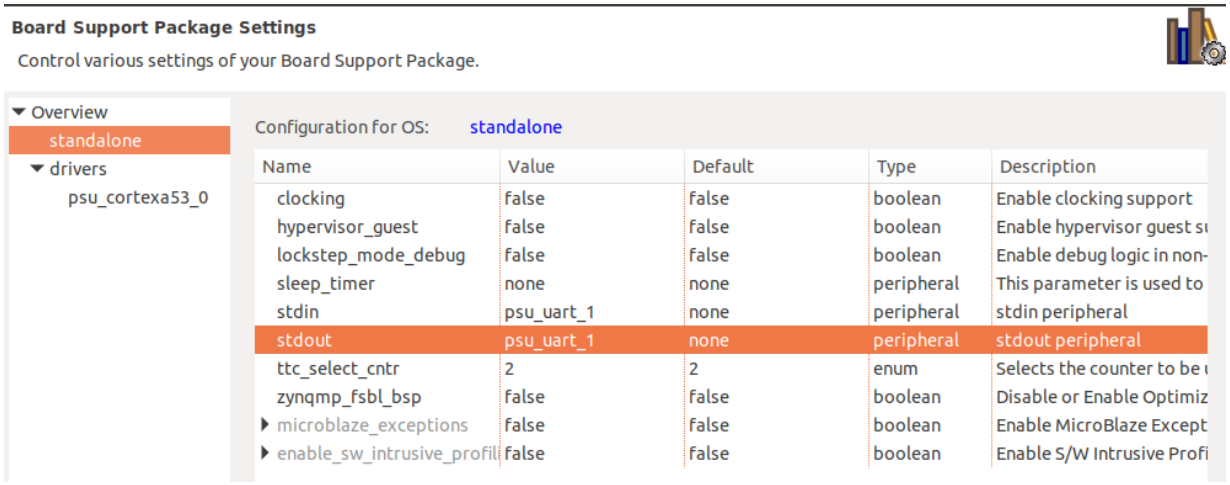
➔ *Aprire il file platform.spr*



➔ *Cliccare su standalone on psu_cortexa53_0 > Board Support Package > Modify BSP Settings*

Esempio
Importazione in Vitis

- ➔ *Cliccare su Overview > standalone;*
- ➔ *Collegare stdin e stdout a psu_uart_1;*
- ➔ *La uart è dispositivo presente all'interno della dev-board che permette la comunicazione seriale tra la board stessa ed il computer host. In questo modo è possibile stampare stringhe sul terminale utilizzando la funzione «printf» del C.*

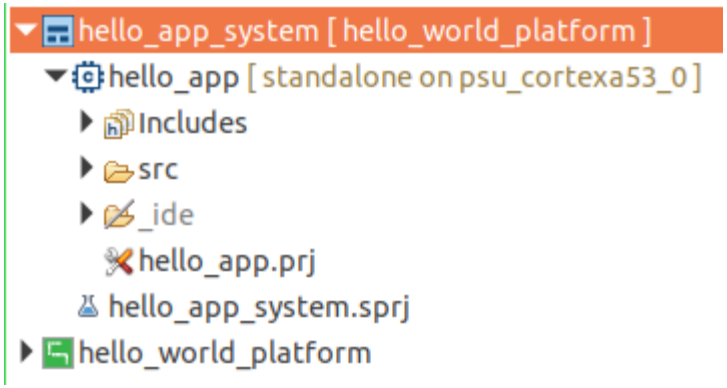


- ➔ *Infine compilare il platform cliccando sull'icona a forma di martello.*

Esempio
Creazione applicazione

- ➔ *Cliccare su New > Application Project;*
- ➔ *Selezionare il platform appena creato;*
- ➔ *Assegnare un nome al progetto;*
- ➔ *Selezionare Standalone on psu_cortexa53_0 come domain;*
- ➔ *Selezionare Hello World come template.*

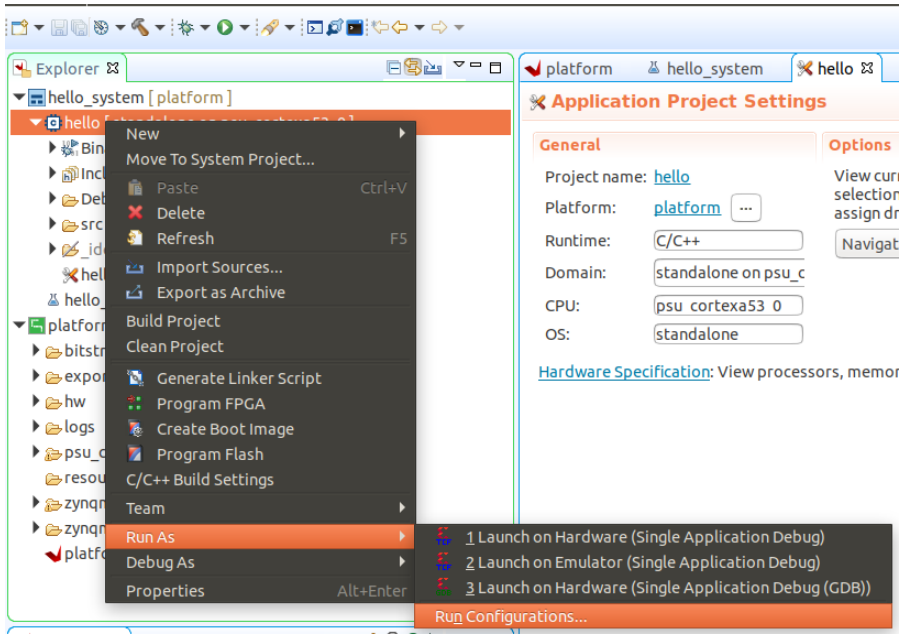
- ➔ *Struttura finale del progetto:*



- ➔ *Cliccare infine sull'icona del martello per compilare anche il progetto software.*

Esempio
Creazione applicazione

➔ *Creare un launch file, cliccando su «Run Configurations»:*



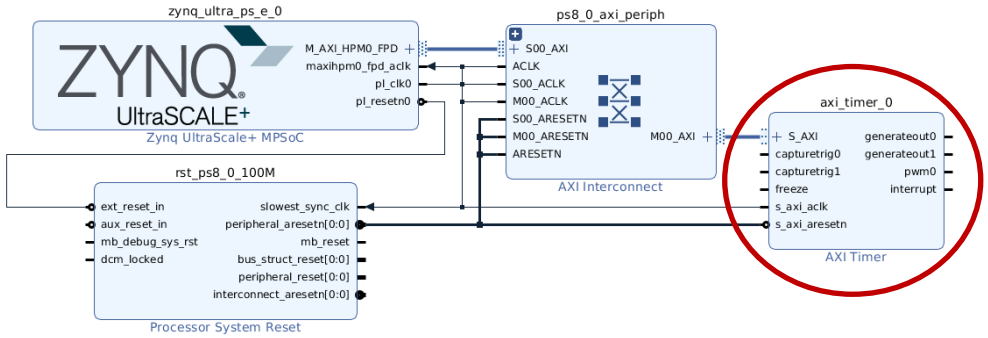
➔ *Doppio click su «Single Application Debug».*

➔ *Assicurarsi che nel parametro «Connection» sia riportato l'hardware server corretto. Fare riferimento al tutorial per l'accesso alle dev-boards.*

Profiling su dev-board

Profiling

➔ *Il profiling di un'applicazione software o acceleratore FPGA, su dev-board può essere svolto ad esempio tramite il timer FPGA collegato all'interno del design:*



➔ *Una volta che viene posizionato il timer nel design, al momento della creazione del platform, Vitis si occupa dell'integrazione delle librerie necessarie all'utilizzo del timer.*

Profiling

```
#include <stdio.h>
#include "platform.h"
#include "xtmrctr.h"
#include "xparameters.h"


int main() {

    init_platform();

    XTmrCtr timer;
    XTmrCtr_Initialize(&timer, XPAR_AXI_TIMER_0_DEVICE_ID);

    XTmrCtr_Start(&timer, 0);
    for(volatile int i = 0; i < 1000; i++) {

    }
    XTmrCtr_Stop(&timer, 0);

    printf("clocks: %d\n", XTmrCtr_GetValue(&timer, 0));

    cleanup_platform();

    return 0;
}
```

- *La funzione XTmrCtr_GetValue ritorna il numero di cicli di clock da quanto il timer è stato avviato.*
- *La precisione del timer dipende dalla frequenza del design;*
- *In questo caso è 10ns.*

Esercizio 01

- *Profilare il tempo di esecuzione di un prodotto di matrici (mmult), implementato in software su Avnet Ultra96, sfruttando l'AXI Timer.*

- *Scaricare i files contenuti sul gitlab del corso (hls/lab2/sw/mmult_sw);*

1. *Prendere nota del numero di cicli di clock impiegati;*
2. *Calcolare il tempo di esecuzione.*

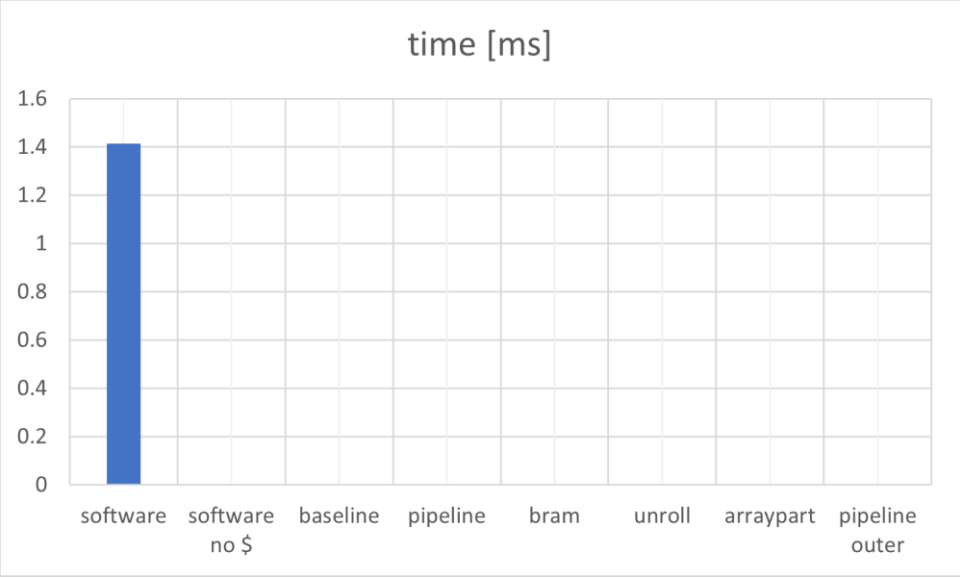
Esercizio 01 - soluzione

```
#define MAX_SIZE 64

void mmult( int *in1,   // Input matrix 1
            int *in2,   // Input matrix 2
            int *out,   // Output matrix (out = A x B)
            int dim     // Size of one dimension of matrix
              )
{
    //Performs matrix multiplication out = in1 x in2
    for (int i = 0; i < dim; i++){
        for (int j = 0; j < dim; j++){
            for (int k = 0; k < dim; k++){
                out[i * dim + j] += in1[i * dim + k] * in2[k * dim  + j];
            }
        }
    }
}
```

```
int main() {

    init_platform();

    XTmrCtr timer;
    XTmrCtr_Initialize(&timer,
XPAR_AXI_TIMER_0_DEVICE_ID);

    int in1[MAX_SIZE*MAX_SIZE];
    int in2[MAX_SIZE*MAX_SIZE];
    int out[MAX_SIZE*MAX_SIZE];

    XTmrCtr_Start(&timer, 0);
    mmult(in1, in2, out, MAX_SIZE);
    XTmrCtr_Stop(&timer, 0);

    printf("clocks: %d\n", XTmrCtr_GetValue(&timer, 0));

    cleanup_platform();

    return 0;
}
```

Esercizio 01 - soluzione

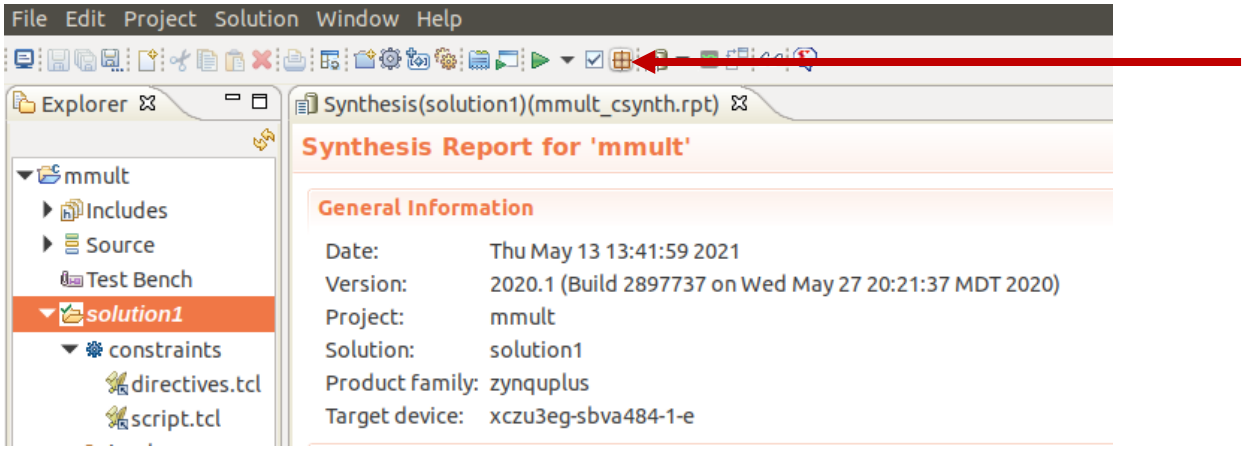
- ➔ 141322 cicli di clock;
- ➔ (141322 clocks / 100000000 Hz)*1000 = **1.413 ms**

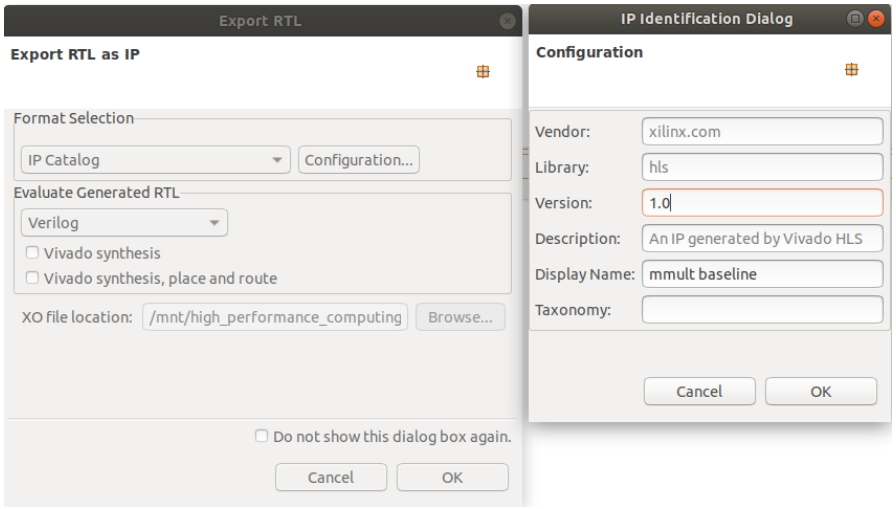


Integrazione di un acceleratore HLS

Export RTL

➔ *Su Vivado HLS dopo aver effettuato una sintesi, esportare l'RTL:*



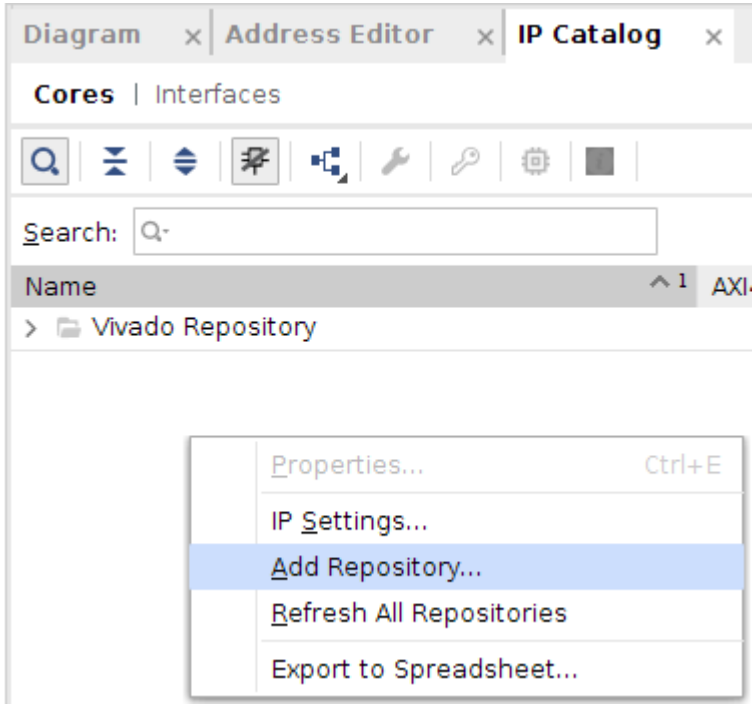


➔ *Una volta esportato l'RTL verrà creato il seguente file all'interno della cartella del progetto HLS:*

➔ *solution1/impl/ip/xilinx_com_hls_mmult_1_0.zip*

Import RTL

➔ *Una volta esportato l'RTL da Vivado HLS, importarlo nel progetto Vivado:*



➔ *Cliccare su IP Catalog > Add Repository;*

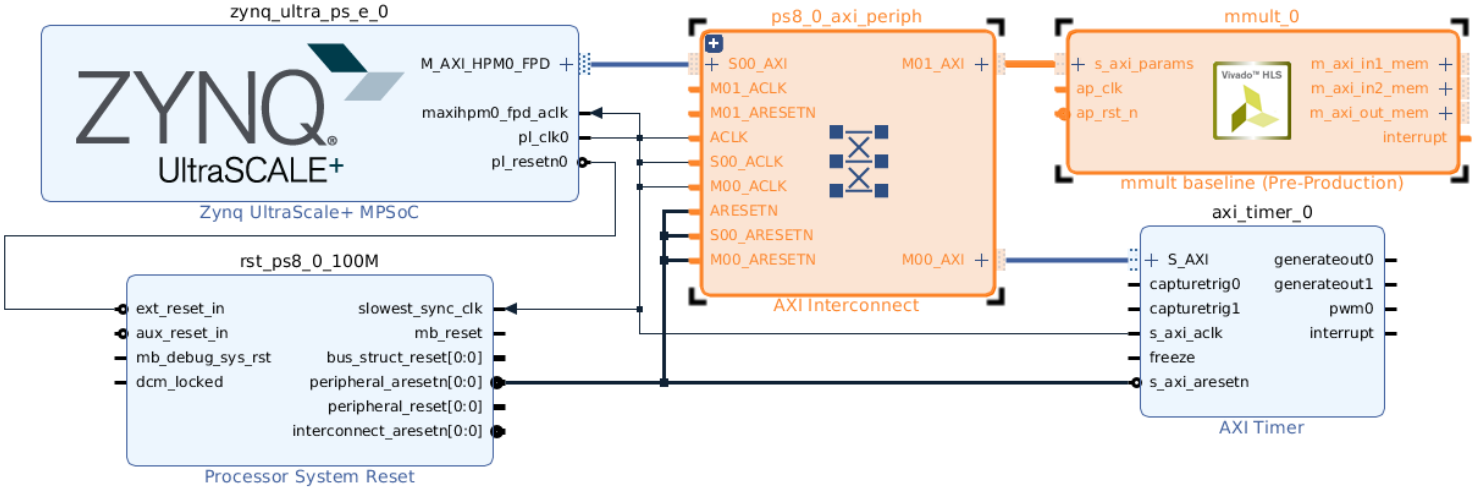
➔ *Selezionare il percorso «solution1/impl/ip/» del progetto Vivado HLS;*

➔ *Una volta importato il pacchetto, aggiungere l'IP al block design.*

Block Design

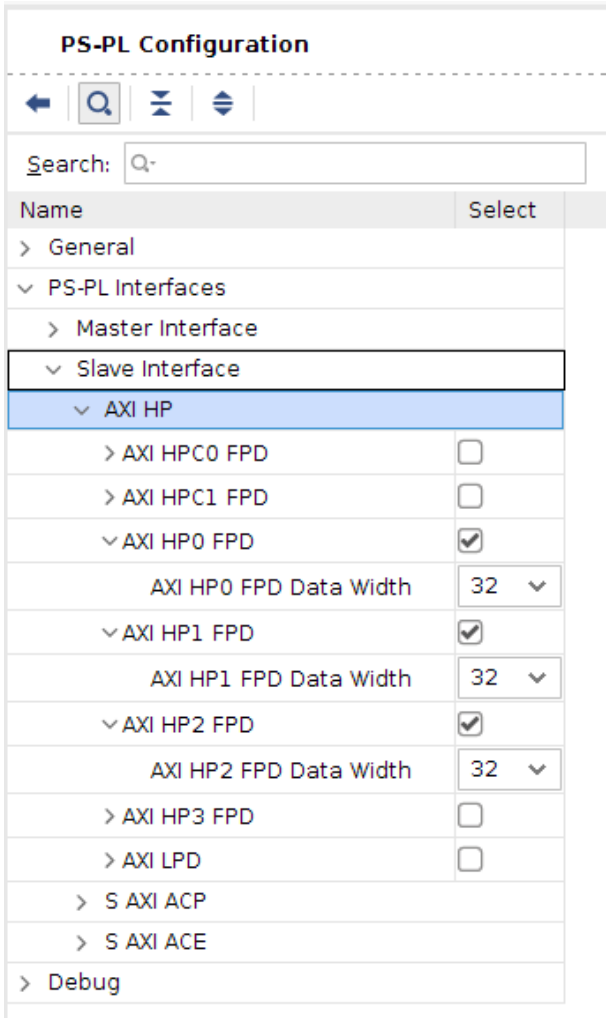
Effettuare i seguenti collegamenti

➔ *porta AXI4-Lite per l'accesso ai registri*



Block Design

Effettuare i seguenti collegamenti

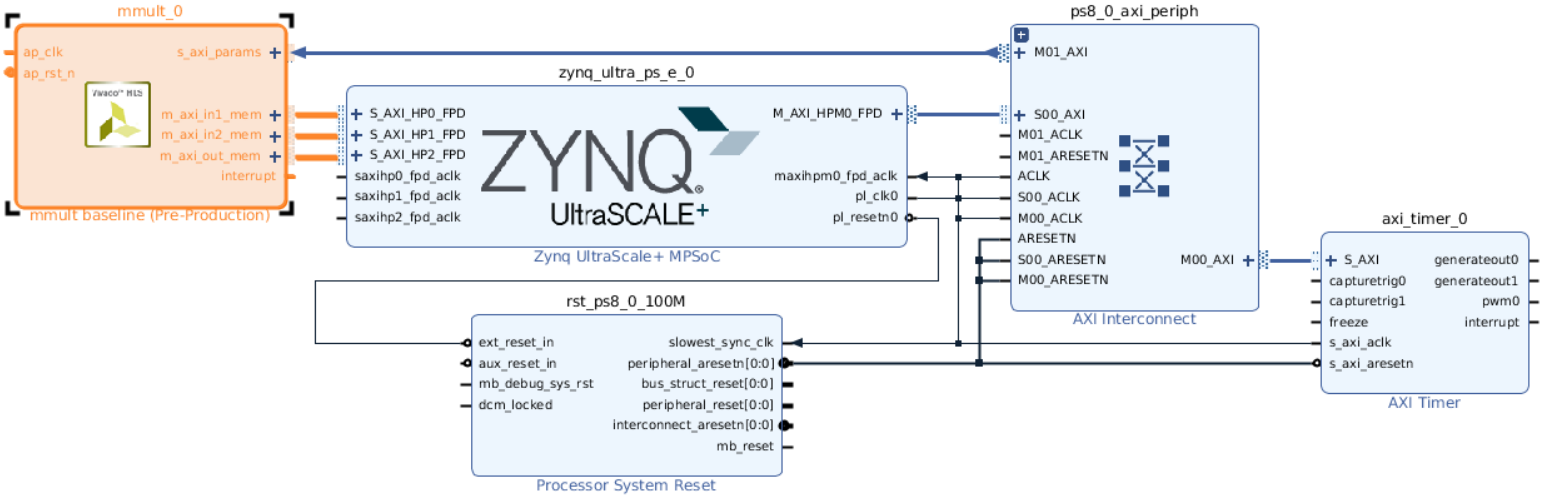


- ➔ *Abilitare tre interfacce AXI4-Slave sul Processing System, una per ciascuna porta m_axi del modulo HLS;*
- ➔ *settare infine la size delle porte a 32 bit;*
- ➔ *Tali porte permettono l'accesso alle matrici, salvate in DRAM.*

Block Design

Effettuare i seguenti collegamenti

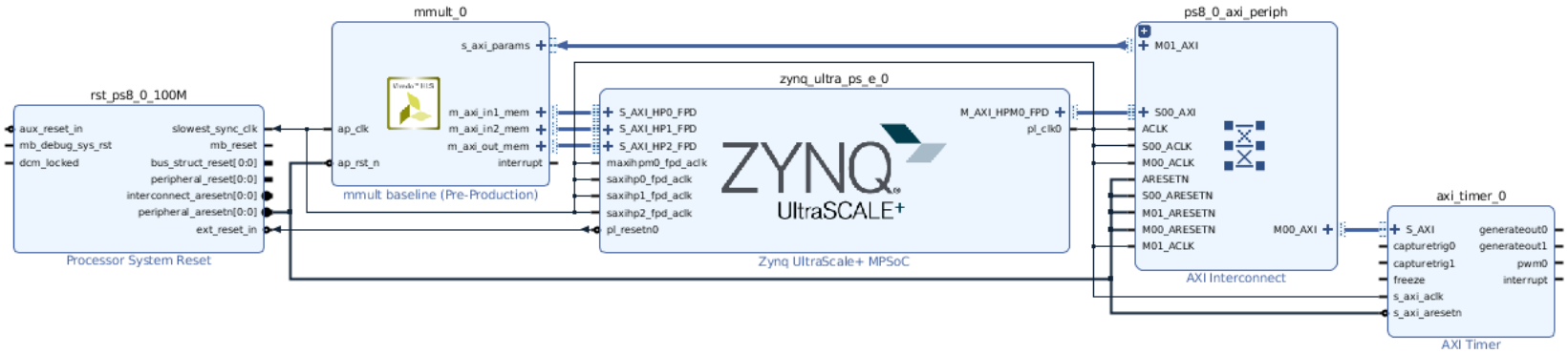
➔ *Porte AXI4 per l'accesso alla DRAM dall'acceleratore FPGA.*



Block Design

Effettuare i seguenti collegamenti

➔ *Cliccare infine su «Run Connection Automation» per collegare gli ingressi di clock ed i reset.*



Block Design
Assegnamento Indirizzi

➔ *Aprire il tab «Address Editor»;*

➔ *Cliccare col destro e selezionare «Assign All». In questo modo Vivado andrà a configurare i segmenti di memoria accessibili dall'acceleratore FPGA. Verranno inoltre impostato l'indirizzo base del register file dell'acceleratore.*

Name	Interface	Slave Segment	Master Base Address	Range	Master High Address
Network 0					
/mmult_0					
/mmult_0/Data_m_axi_in1_mem (32 address bits : 4G)					
/zynq_ultra_ps_e_0	S_AXI_HP...	HP0_DDR_LOW	0x0000_0000	2G	0x7FFF_FFFF
Excluded (1)					
/mmult_0/Data_m_axi_in2_mem (32 address bits : 4G)					
/zynq_ultra_ps_e_0	S_AXI_HP...	HP1_DDR_LOW	0x0000_0000	2G	0x7FFF_FFFF
Excluded (1)					
/mmult_0/Data_m_axi_out_mem (32 address bits : 4G)					
/zynq_ultra_ps_e_0	S_AXI_HP...	HP2_DDR_LOW	0x0000_0000	2G	0x7FFF_FFFF
Excluded (1)					
/zynq_ultra_ps_e_0					
/zynq_ultra_ps_e_0/Data (39 address bits : 0x00A0000000 [256M] ,0x0400000000 [4G] ,0x1000000000 [224G])					
/axi_timer_0	S_AXI	Reg	0x00_A000_0000	64K	0x00_A000_FFFF
/mmult_0	s_axi_par...	Reg	0x00_A001_0000	64K	0x00_A001_FFFF

Block Design
Validation

- *Cliccare infine su «Validate Design»;*
- *Se tutto è configurato bene dovrebbe comparire la seguente schermata:*

- *Infine generare il bitstream.*
- *In questo caso, partendo dal design già sintetizzato nell'esempio precedente, parte del design stesso non necessita sintesi ulteriori, in questo modo i tempi di generazione del bitstream sono ridotti.*

Configurazione di un acceleratore HLS

Configurazione IP HLS

➔ *Come visto nell'esempio precedente, dopo aver:*

1. *generato il bitstream;*
2. *Esportato il file .xsa contenente la descrizione dell'hardware;*

➔ *Avviare Vitis ed importare il file .xsa con la nuova descrizione dell'hardware all'interno di un nuovo platform.*

Configurazione IP HLS

- *La configurazione dell'acceleratore avviene in maniera molto simile a quanto visto per l'AXI Timer.*
- *In particolare Vivado HLS genera un file chiamato:*
  - *x<nome_top_function>.h*
- *Contenente i prototipi di tutte le funzioni necessarie all'interazione con l'acceleratore.*

```
int XMmult_Initialize(XMmult *InstancePtr, u16 DeviceId);
```

- *Inizializzazione dell'acceleratore*

Configurazione IP HLS

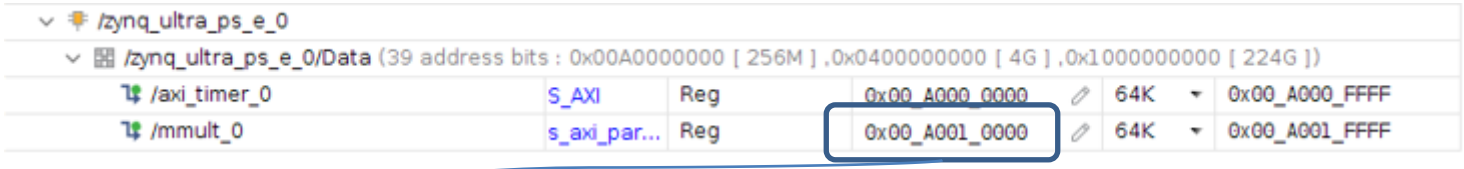
```
void XMmult_Start(XMmult *InstancePtr);
u32 XMmult_IsDone(XMmult *InstancePtr);
u32 XMmult_IsIdle(XMmult *InstancePtr);
u32 XMmult_IsReady(XMmult *InstancePtr);
void XMmult_EnableAutoRestart(XMmult *InstancePtr);
void XMmult_DisableAutoRestart(XMmult *InstancePtr);
```

➔ *Configurazione dei segnali di controllo.*

```
void XMmult_Set_in1(XMmult *InstancePtr, u32 Data);
u32 XMmult_Get_in1(XMmult *InstancePtr);
void XMmult_Set_in2(XMmult *InstancePtr, u32 Data);
u32 XMmult_Get_in2(XMmult *InstancePtr);
void XMmult_Set_out_r(XMmult *InstancePtr, u32 Data);
u32 XMmult_Get_out_r(XMmult *InstancePtr);
void XMmult_Set_dim(XMmult *InstancePtr, u32 Data);
u32 XMmult_Get_dim(XMmult *InstancePtr);
```

➔ *Scrittura e lettura del register file dell'accleratore.*

Configurazione IP HLS



➔ *Indirizzo base*

```
// 0x00 : Control signals
//        bit 0  - ap_start (Read/Write/COH)
//        bit 1  - ap_done (Read/COR)
//        bit 2  - ap_idle (Read)
//        bit 3  - ap_ready (Read)
//        bit 7  - auto_restart (Read/Write)
//        others - reserved
// 0x10 : Data signal of in1
//        bit 31~0 - in1[31:0] (Read/Write)
// 0x18 : Data signal of in2
//        bit 31~0 - in2[31:0] (Read/Write)
// 0x20 : Data signal of out_r
//        bit 31~0 - out_r[31:0] (Read/Write)
// 0x28 : Data signal of dim
//        bit 31~0 - dim[31:0] (Read/Write)
```

```
#define XMMULT_PARAMS_ADDR_AP_CTRL      0x00
#define XMMULT_PARAMS_ADDR_GIE          0x04
#define XMMULT_PARAMS_ADDR_IER          0x08
#define XMMULT_PARAMS_ADDR_ISR          0x0c
#define XMMULT_PARAMS_ADDR_IN1_DATA     0x10
#define XMMULT_PARAMS_BITS_IN1_DATA     32
#define XMMULT_PARAMS_ADDR_IN2_DATA     0x18
#define XMMULT_PARAMS_BITS_IN2_DATA     32
#define XMMULT_PARAMS_ADDR_OUT_R_DATA   0x20
#define XMMULT_PARAMS_BITS_OUT_R_DATA   32
#define XMMULT_PARAMS_ADDR_DIM_DATA     0x28
#define XMMULT_PARAMS_BITS_DIM_DATA     32
```

➔ *Offsets dei singoli registri interni all'acceleratore*

Configurazione IP HLS
Esempio

➔ *Nell'esempio della Mmult supponendo di voler settare il registro relativo alla dimensione delle matrici:*

```
XMmult_Set_dim(InstancePtr, dim);
```

➔ *Che equivale a:*

```
*((volatile unsigned int*)(0xA0010000 + XMMULT_PARAMS_ADDR_DIM_DATA)) = dim;
```

➔ *Indirizzo base, impostato tramite l'address editor di Vivado.*

➔ *Offset generato da Vivado HLS.*

Codice di esempio

```
void mmult_hardware(
        int *in1,   // Input matrix 1
        int *in2,   // Input matrix 2
        int *out,   // Output matrix (out = A x B)
        int dim     // Size of one dimension of matrix
                                        ) {

                    Xil_DCacheFlush();

                    XMmult mmult_accel;
                    XMmult_Initialize(&mmult_accel, XPAR_MMULT_0_DEVICE_ID);

                    XMmult_Set_in1(&mmult_accel, in1);
                    XMmult_Set_in2(&mmult_accel, in2);
                    XMmult_Set_out_r(&mmult_accel, out);
                    XMmult_Set_dim(&mmult_accel, MAX_SIZE);

                    XMmult_Start(&mmult_accel);

                    while(!XMmult_IsDone(&mmult_accel)){
                                        /* wait polling */
                    }
                    Xil_DCacheInvalidate();
}
```

```
int main() {

    init_platform();

    …

    XTmrCtr_Start(&timer, 0);
    mmult_hardware(in1, in2, out, MAX_SIZE);
    XTmrCtr_Stop(&timer, 0);

    …

    cleanup_platform();

    return 0;
}
```

➔ *Prima e dopo l'offloading sull'acceleratore è necessario un flush ed invalidate della Data Cache. Questo perché l'acceleratore non è coherent rispetto alla CPU, ma accede direttamente in DRAM.*

Esercizi

➔ *Implementare le seguenti versioni dell'algoritmo mmult e testare su dev-board Ultra96;*

➔ *Svolgere i seguenti esperimenti:*

- ***(exercise_2)** Mmult baseline, senza ottimizzazioni;*
- ***(exercise_3)** Mmult pipelined;*
- ***(exercise_4)** Mmult con Blocked RAM;*
- ***(exercise_5)** Mmult Unrolled;*
- ***(exercise_6)** Mmult con Array Partition;*
- ***(exercise_7)** Mmult fully pipelined.*

- *Calcolare lo Speedup ottenuto per ciascun esercizio rispetto alla versione implementata in software.*

➔ ***NB**: scaricare da gitlab il software di controllo per l'acceleratore Mmult (hls/lab2/sw/).*

Soluzioni

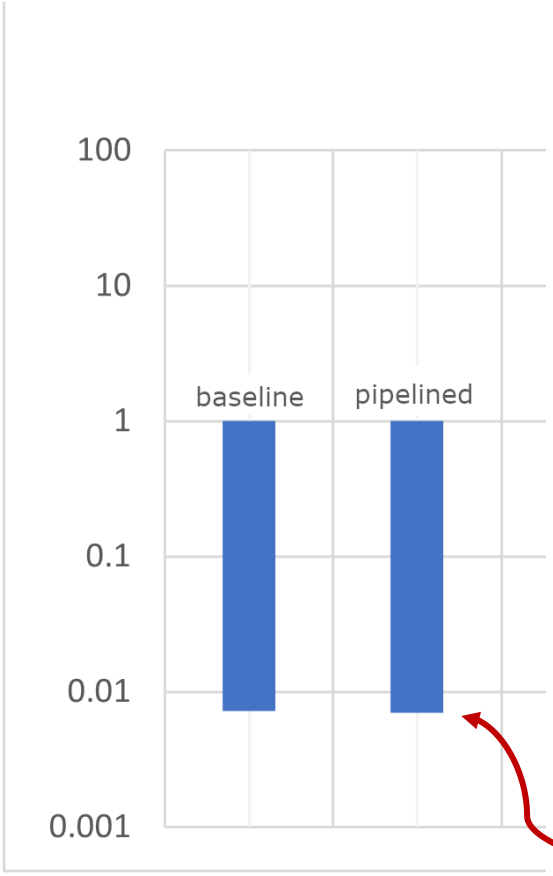
- *Presenti sul gitlab del corso all'interno della cartella hls/lab2.*
  - *Acceleratori HLS (hls/lab2/exported_ips/):*

    xilinx_com_hls_exercise_xy
  - *Designs Vivado da sintetizzare in formato .tcl (hls/lab2/):*

    exercise_xy.tcl
  - *Designs Vivado sintetizzati in formato .xsa (hls/lab2/hw/):*

    exercise_xy.xsa

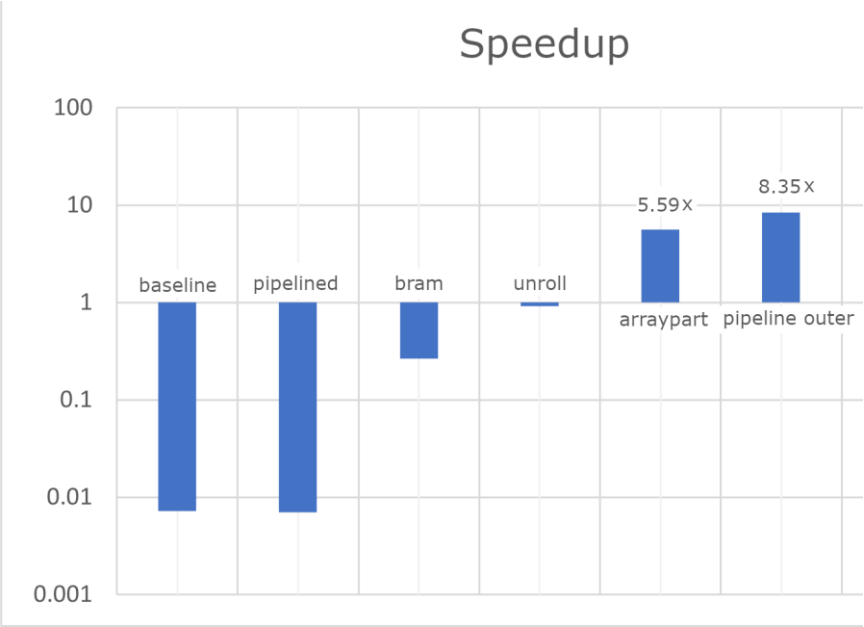
Soluzioni
exercise_2/3



- ➔ *La versione baseline e pipelined, rispetto all'implementazione software comportano uno slowdown di circa 140x.*
- ➔ *Principalmente a causa di due aspetti:*
  - ➔ *Hardware sequenziale (baseline), che lavora a bassa frequenza, rispetto alla CPU: 100MHz FPGA e 1.3GHz Cortex A53.*
  - ➔ *Accesso diretto in DRAM, senza cache o scratchpad.*
- ➔ *In questo caso l'incremento sul throughput non comporta nessun beneficio.*

Soluzioni

exercise_4/5/6/7



➔ *Notiamo che il distacco prestazionale più importante si ha con l'introduzione della BRAM e dell'Array Patitioning.*

➔ *Lo speedup finale ottenuto è di **8.35x** rispetto alla versione software.*

Tuning della frequenza

Tuning della frequenza

➔ *Come ulteriore incremento prestazionale, è possibile anche incrementare leggermente la frequenza dell'acceleratore. Per l'incremento di frequenza è necessario svolgere i seguenti step:*

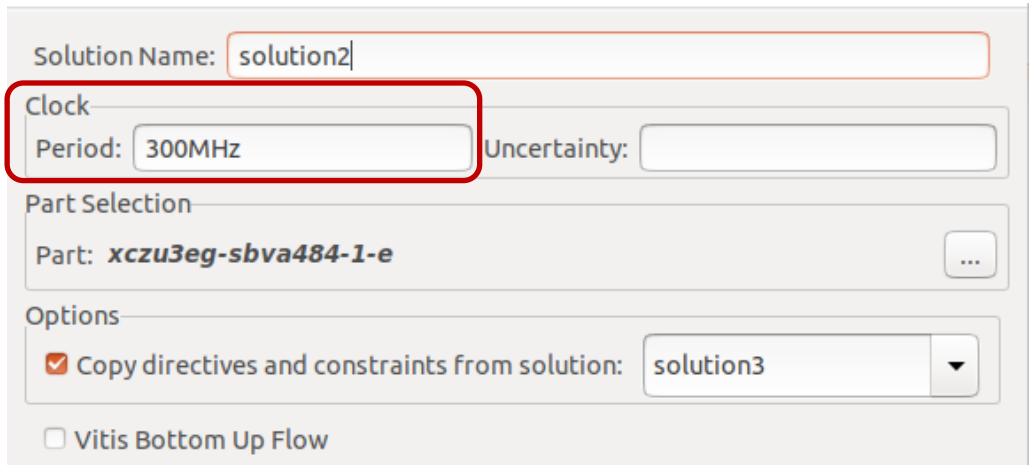
1. *Generare un acceleratore HLS inserendo come target frequenza/periodo desiderati;*

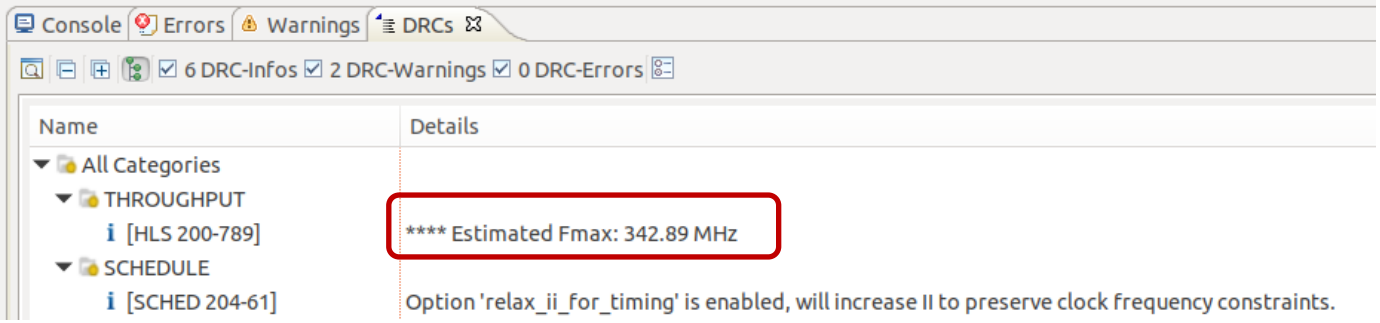
2. *Assicurarsi che Vivado HLS riporti come «Estimated Fmax» una frequenza maggiore o uguale rispetto alla frequenza desiderata.*

3. *Collegare l'uscita di clock del PS al componente Clocking Wizard e selezionare la frequenza desiderata.*

Tuning della frequenza

Step 1 / 2 - Esempio

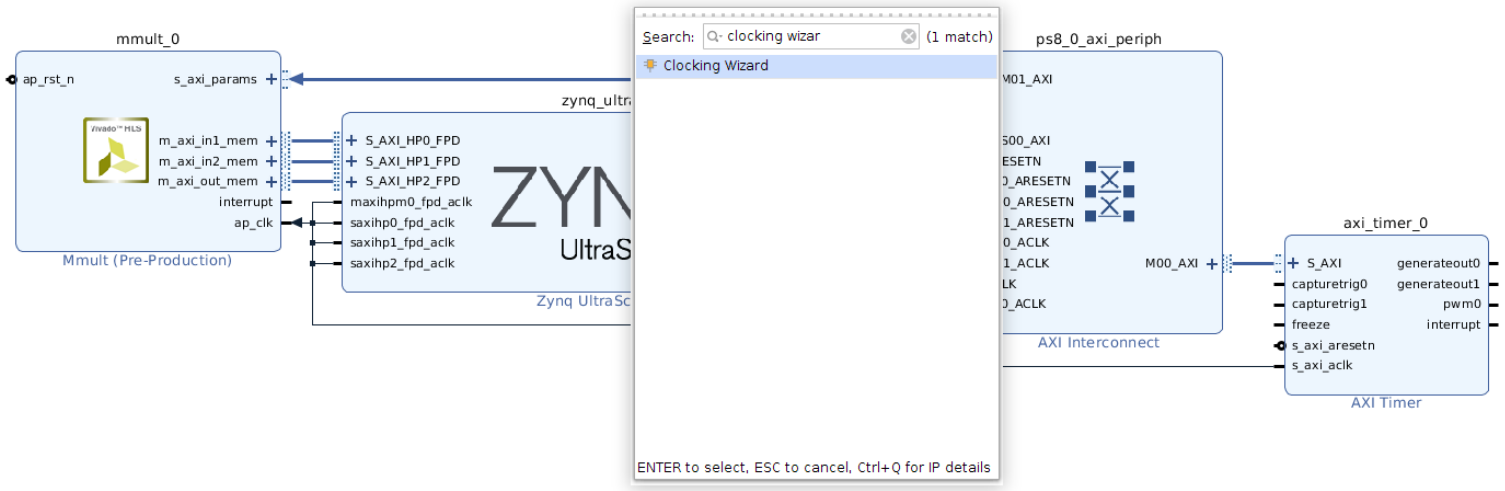




Tuning della frequenza

Step 3- Esempio

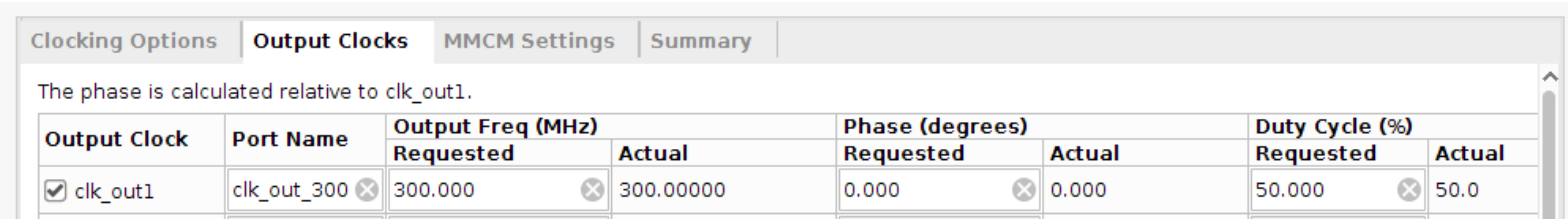
➔ *Aggiungere il componente «Clocking Wizard» al design.*



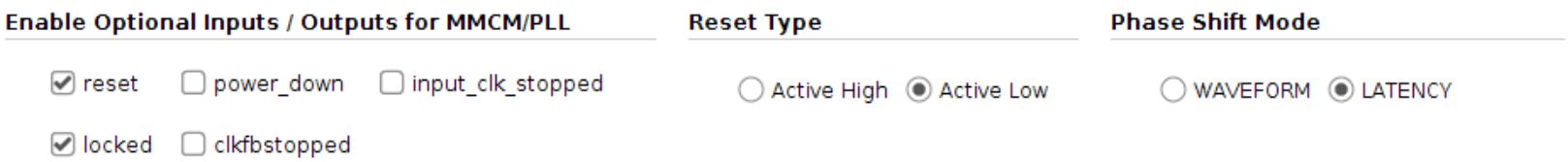
Tuning della frequenza

Step 3- Esempio

➔ *Settare la frequenza desiderata su «Output Freq». Eventualmente rinominare anche la porta, ad esempio: «clk_out_300».*



➔ *Impostare il reset su attivo basso.*

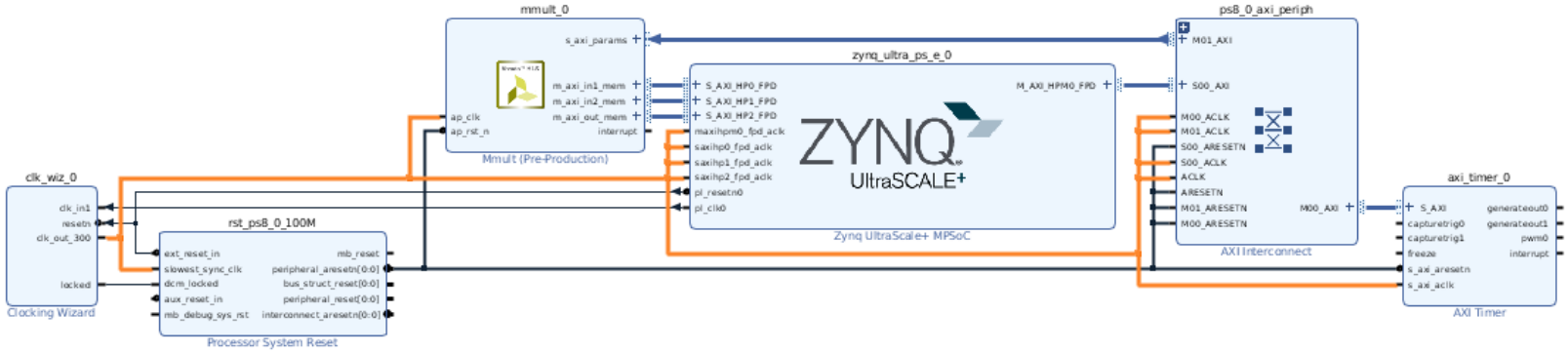


Tuning della frequenza

Step 3- Esempio

➔ *Collegare tutti gli ingressi di clock dei componenti all'uscita del Wizard «clk_out_300»;*

➔ *Portare l'uscita di clock del PS verso l'ingresso di clock del Wizard.*



Esercizi

➔ *Partire dall'acceleratore dell'exercise_7, Mmult fully pipelined e provare ad incrementare la frequenza.*

➔ *Svolgere i seguenti esperimenti:*

- ***(exercise_8)** Mmult pipelined 200MHz;*
- ***(exercise_9)** Mmult pipelined 300MHz;*

- *Calcolare lo Speedup ottenuto per ciascun esercizio rispetto alla versione implementata in software.*

➔ ***NB**: scaricare da gitlab il software di controllo per l'acceleratore Mmult (hls/lab2/sw/).*

Soluzioni

- ➔ *In questo caso è stato possibile incrementare la frequenza dell'acceleratore fino ad un massimo di 300MHz.*
- ➔ *Lo speedup finale ottenuto è pari a **23.8x**.*