

1 Guide FPGA

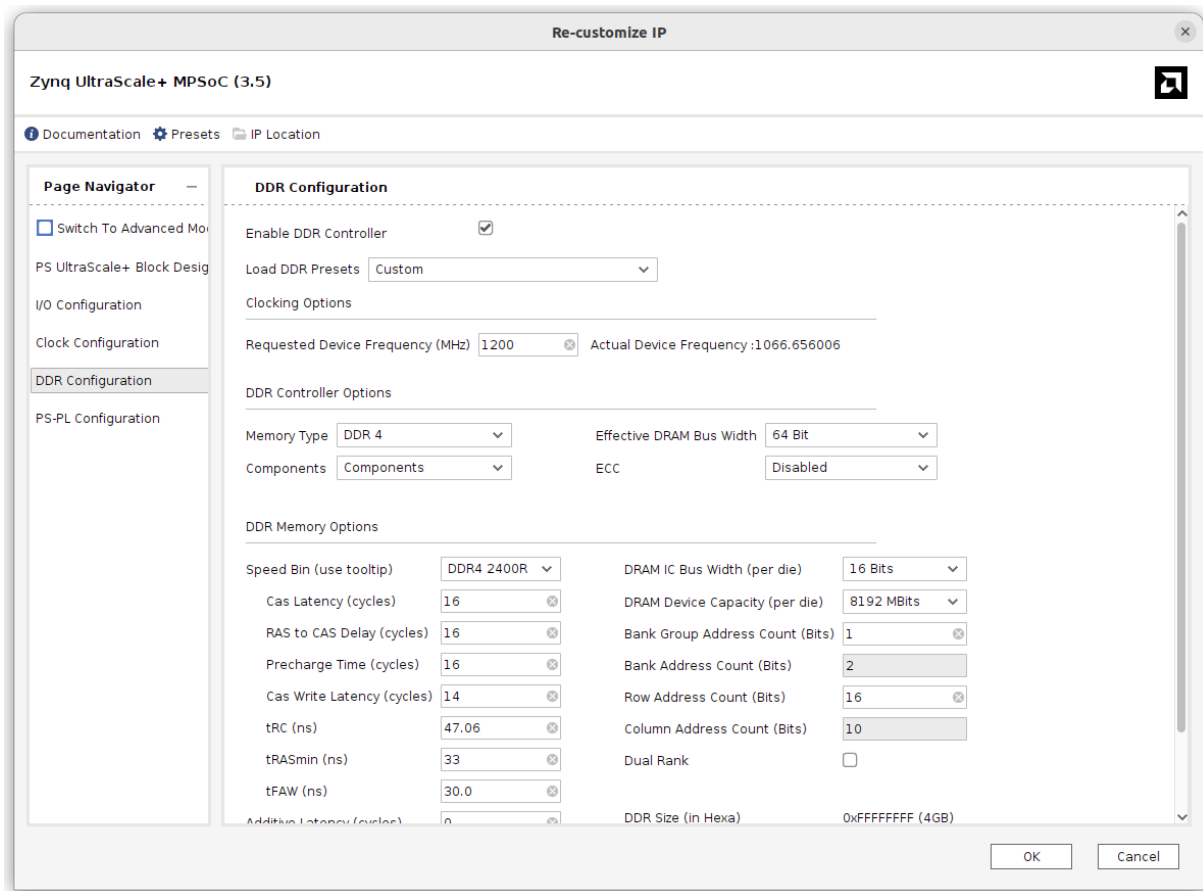
Aides des dossiers suivants :

- AXI GPIO documentation ABTICS
- AXI STREAM documentation ABTICS
- AXI UARTLITE documentation ABTICS

2 Vivado

2.1 Fonctionnement de la carte Kria

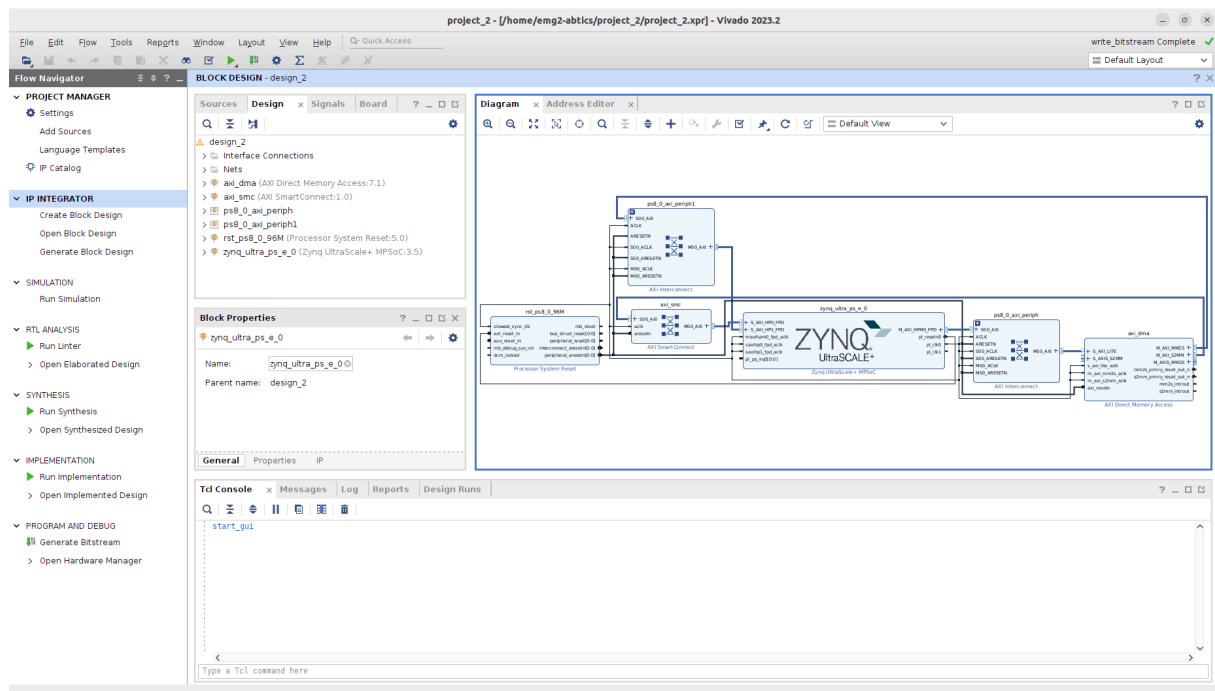
Mise en place d'une documentation explicitant les changements nécessaires à faire pour faire fonctionner la carte AMD Xilinx Kria KV260 disponible dans le fichier suivant : [zynq_modification.pdf](#). Ces modifications sont à faire sur l'IP Zynq Ultrascale +.



3 Design DMA

3.1 Vivado

Dans cette partie, nous avons mis en place un design permettant de faire l'envoi et la réception d'une donnée de taille 32 bits grace aux DMA.



3.2 Vitis classic

Fonctionne pas mais code accessible dans `project_2`

```

#include "xparameters.h"
#include "xaxidma.h"
#include "xil_printf.h"
#include "xil_cache.h"

#define DMA_DEV_ID    XPAR_AXIDMA_0_DEVICE_ID
#define DDR_BASE_ADDR XPAR_PSU_DDR_0_S_AXI_BASEADDR
#define MEM_BASE_ADDR (DDR_BASE_ADDR + 0xA000000)

#define TX_BUFFER_BASE (MEM_BASE_ADDR + 0x00100000)
#define RX_BUFFER_BASE (MEM_BASE_ADDR + 0x00300000)
#define RX_BUFFER_HIGH (MEM_BASE_ADDR + 0x004FFFFFF)

#define MAX_PKT_LEN    0x20
#define NUMBER_OF_TRANSFERS 10

XAxidma AxiDma;

int CheckData() {
    u32 *TxBufferPtr = (u32 *)TX_BUFFER_BASE;
    u32 *RxBufferPtr = (u32 *)RX_BUFFER_BASE;
    u32 Index;

    for (Index = 0; Index < MAX_PKT_LEN; Index++) {
        TxBufferPtr[Index] = Index + 0xC;
        RxBufferPtr[Index] = 0;
    }

    Xil_DCacheFlushRange((UINTPTR)TxBufferPtr, MAX_PKT_LEN * sizeof(u32));
    // Send a packet
    int Status = XAxidma_SimpleTransfer(&AxiDma, (UINTPTR)RxBufferPtr, MAX_PKT_LEN
* sizeof(u32), XAXIDMA_DEVICE_TO_DMA);
    if (Status != XST_SUCCESS) {
        return XST_FAILURE;
    }
    Status = XAxidma_SimpleTransfer(&AxiDma, (UINTPTR)TxBufferPtr, MAX_PKT_LEN *
sizeof(u32), XAXIDMA_DMA_TO_DEVICE);
    if (Status != XST_SUCCESS) {
        return XST_FAILURE;
    }
    while ((XAxidma_Busy(&AxiDma, XAXIDMA_DMA_TO_DEVICE)) || (XAxidma_Busy(&AxiDma,
XAXIDMA_DEVICE_TO_DMA))) {
        // Wait
    }
    // Invalidate the cache to receive the data
    Xil_DCacheInvalidateRange((UINTPTR)RxBufferPtr, MAX_PKT_LEN * sizeof(u32));
    for (Index = 0; Index < MAX_PKT_LEN; Index++) {
        if (RxBufferPtr[Index] != TxBufferPtr[Index]) {
            xil_printf("Data error %d: %x/%x\r\n", Index, (unsigned
int)RxBufferPtr[Index], (unsigned int)TxBufferPtr[Index]);
            return XST_FAILURE;
        }
    }
    return XST_SUCCESS;
}

```

```

int main() {
    int Status;
    XAxiDma_Config *Config;
    xil_printf("init\n");

    // Initialize the XAxiDma device.
    Config = XAxiDma_LookupConfig(DMA_DEV_ID);
    if (!Config) {
        xil_printf("No config found for %d\r\n", DMA_DEV_ID);
        return XST_FAILURE;
    }

    Status = XAxiDma_CfgInitialize(&AxiDma, Config);
    if (Status != XST_SUCCESS) {
        xil_printf("Initialization failed %d\r\n", Status);
        return XST_FAILURE;
    }

    if (XAxiDma_HasSg(&AxiDma)) {
        xil_printf("Device configured as SG mode\r\n");
        return XST_FAILURE;
    }

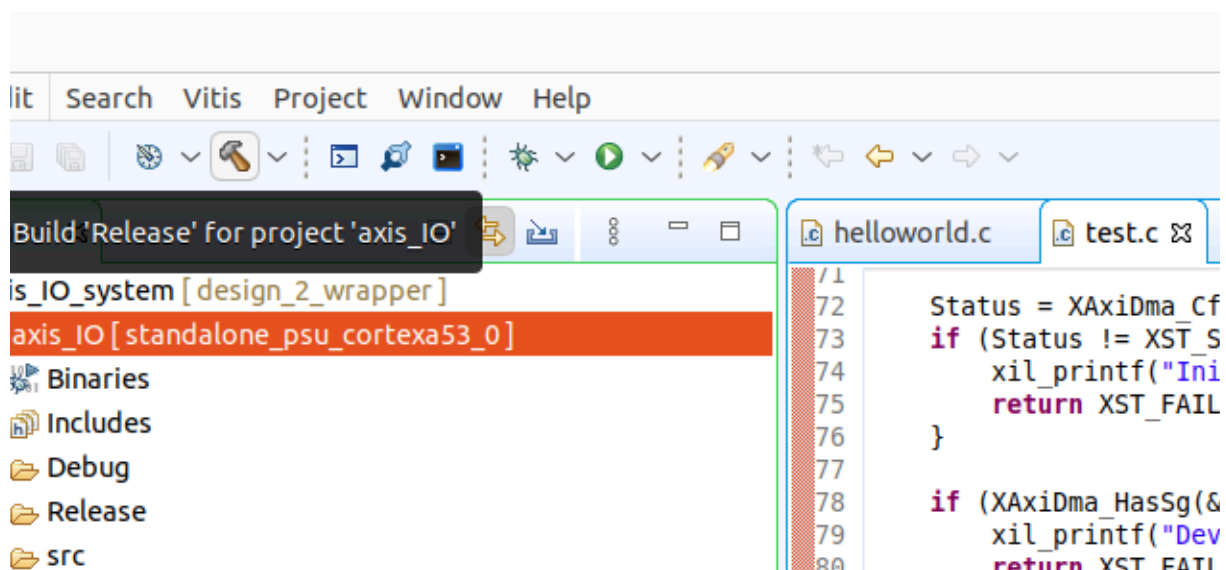
    // Disable interrupts, we use polling mode.
    XAxiDma_IntrDisable(&AxiDma, XAXIDMA_IRQ_ALL_MASK, XAXIDMA_DEVICE_TO_DMA);
    XAxiDma_IntrDisable(&AxiDma, XAXIDMA_IRQ_ALL_MASK, XAXIDMA_DMA_TO_DEVICE);

    // Transmit and receive the data.
    Status = CheckData();

    if (Status != XST_SUCCESS) {
        xil_printf("Failed data check\r\n");
        return XST_FAILURE;
    }

    xil_printf("Successfully ran AXI DMA example\r\n");
    return XST_SUCCESS;
}

```



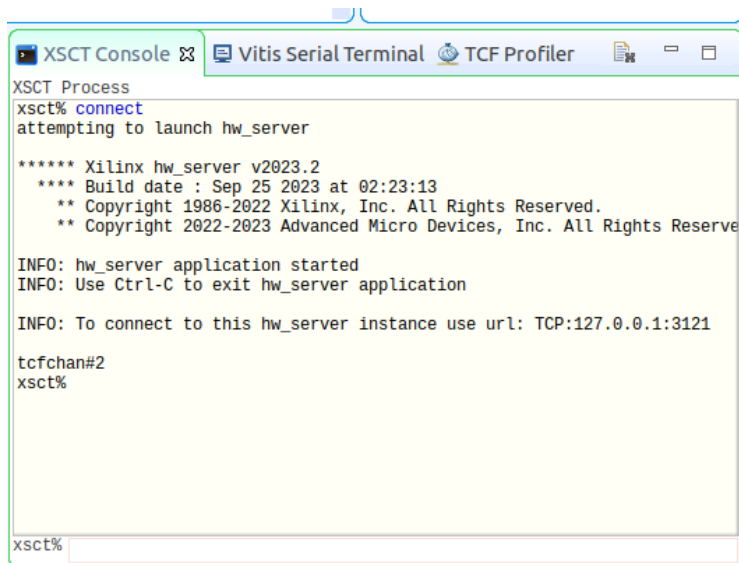
3.3 Connexion FPGA

Il faut connecter le FPGA et lancer les commandes suivantes dans le **XSCT Console** :

Test du design :

- Test envoie une donnée d'une adresse à une autre :
 - initialisation du fpga :

```
connect
targets -set -filter {name =~ "PSU"}
mwr 0xffca0010 0x0
mwr 0xff5e0200 0x0100
rst -system
```
- Run programme :



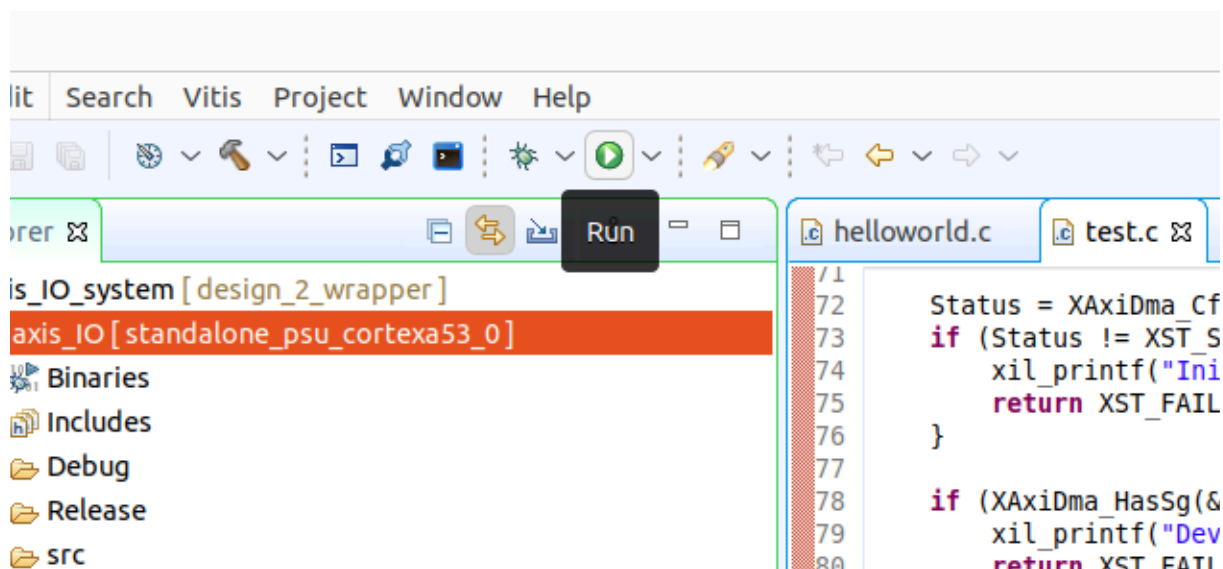
```
XSCT Console | Vitis Serial Terminal | TCF Profiler
XSCT Process
xsct% connect
attempting to launch hw_server

***** Xilinx hw_server v2023.2
**** Build date : Sep 25 2023 at 02:23:13
** Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.
** Copyright 2022-2023 Advanced Micro Devices, Inc. All Rights Reserved

INFO: hw_server application started
INFO: Use Ctrl-C to exit hw_server application

INFO: To connect to this hw_server instance use url: TCP:127.0.0.1:3121

tcfchan#2
xsct%
```



4 Vitis HLS

Mise en place d'un code d'addition de 2 float format 32 bits simple avec un interfacement axi pour Vivado.

```

#include "stdio.h"
#include "add.h"

void add(data_h in_a, data_h in_b, data_h out_c)
{
#pragma HLS INTERFACE ap_ctrl_none port=return
#pragma HLS INTERFACE s_axilite port=in_a
#pragma HLS INTERFACE s_axilite port=in_b
#pragma HLS INTERFACE s_axilite port=out_c

    out_c = in_a + in_b;
}

```

```

#ifndef _MY_ADD_H_
#define _MY_ADD_H_

typedef float data_h;

void add(data_h in_a, data_h in_b, data_h out_c);

#endif

```

Dans Vitis HLS, il est indispensable, pour que la génération du code VHDL, de faire un test.

```

#include "stdio.h"
#include "add.h"

int main() {
    data_h a,b,c;

    a = 1.0;
    b = 2.0;

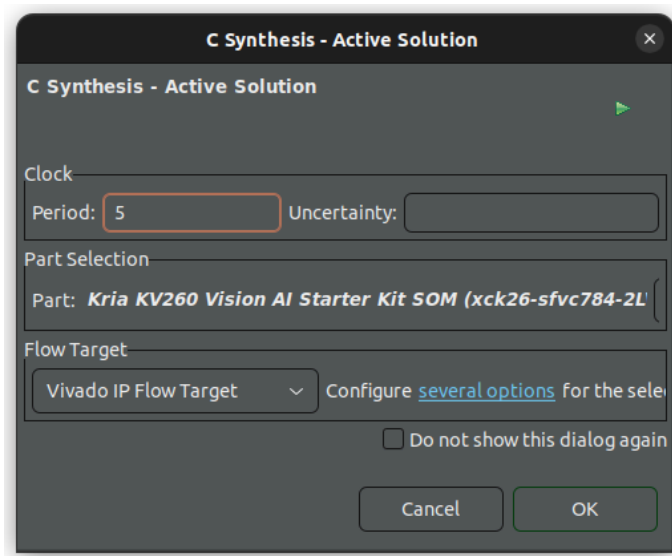
    add(a, b, c);
    if (c == 3.0){
        printf("test passed !\n");
        return 0;
    }
    return 1;
}

```

Afin de pouvoir obtenir une IP compilée il faut valider plusieurs étapes dans Vitis HLS :

- La synthèse :

Les paramètres mis :



Le résultat obtenu :

Synthesis Summary Report of 'add'

General Information

Date: Wed Sep 11 15:17:38 2024
Version: 2023.2 (Build 4023990 on Oct 11 2023)
Project: posit_hls

Solution: solution1 (Vivado IP Flow Target)
Product family: zynqplus
Target device: xck26-sfvc784-2LV-c

Timing Estimate

Target	Estimated	Uncertainty
5.00 ns	0 ns	1.35 ns

Performance & Resource Estimates

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM	
@ add					0	0.0			1		no	0	0	144	232	0

Performance Pragma

N/A

HW Interfaces

S_AXILITE Interfaces

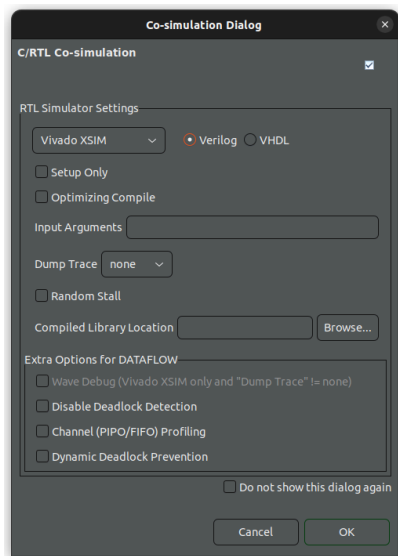
Interface	Data Width	Address Width	Offset	Register
s_axi_control	32	6	16	0

S_AXILITE Registers

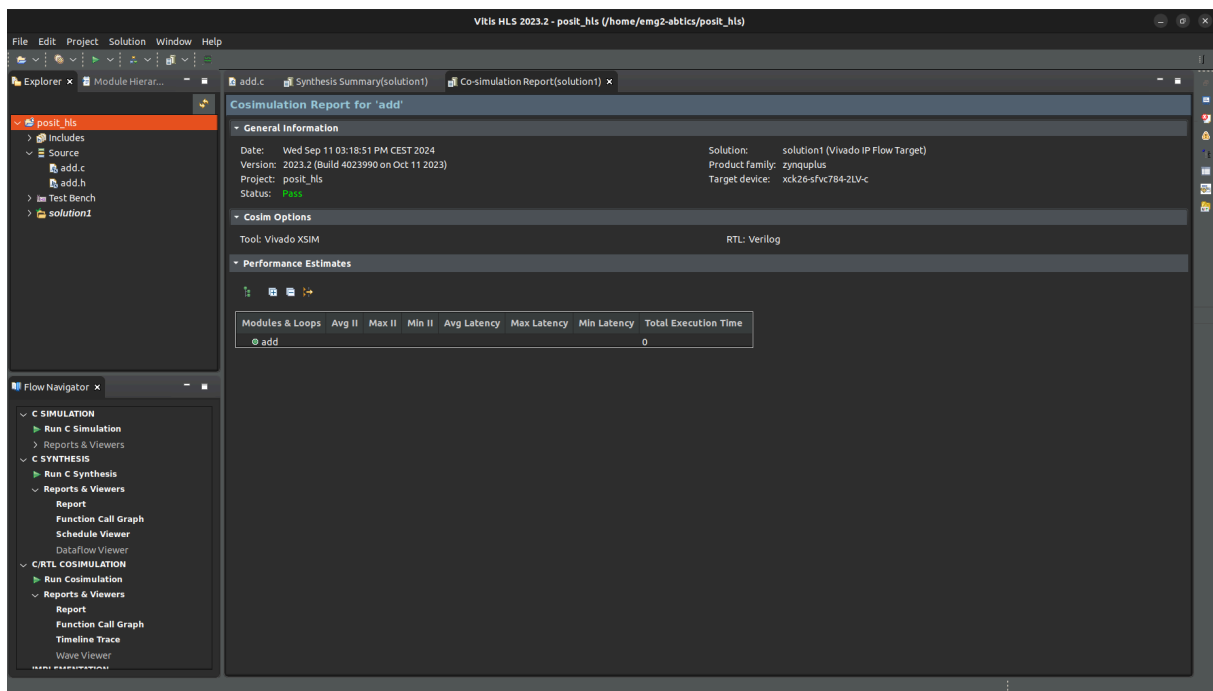
Interface	Register	Offset	Width	Access	Description
s_axi_control	in_a	0x10	32	W	Data signal of in_a
s_axi_control	in_b	0x18	32	W	Data signal of in_b
s_axi_control	out_c	0x20	32	W	Data signal of out_c

- La simulation :

Les paramètres mis :

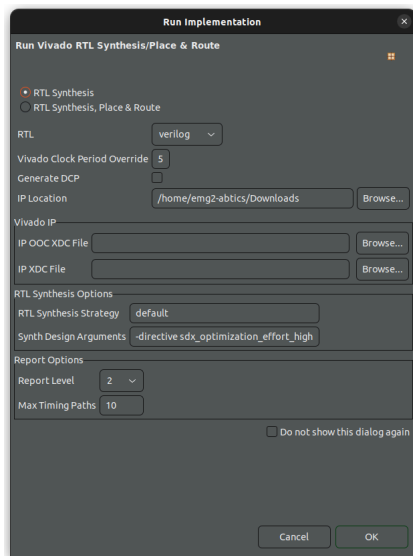


Le résultat obtenu :



- L'implémentation :

Les paramètres mis :



Le résultat obtenu :

Export Report for 'add'

General Information

Report date: Wed Sep 11 15:23:21 CEST 2024
 Project: posit_hls
 Solution: solution1
 Device target: xc7z020clg484-2LVC
 Implementation tool: Xilinx Vivado v.2023.2

Run Constraints & Options

Name	Value
> Design Constraints & Options	
> RTL Synthesis Options	
> Reporting Options	

Resource Usage

	Verilog
SLICE	0
LUT	94
FF	139
DSP	0
BRAM	0
URAM	0
LATCH	0
SRL	0
CLB	0

Final Timing

	Verilog
CP required	3
CP achieved post-synthesis	NA

No Sequential Path

Resources

Name	LUT	FF	DSP	BRAM	URAM	SRL	Pragma	Impl	Latency	Variable	Source
> inst	94	139									
> control_s_axi_U	94	139									

Fail Fast