FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Circuitos Logicos Programables
Carrera de especialización sistemas embebidos
ING.Jose Mauricio Lara Tapia

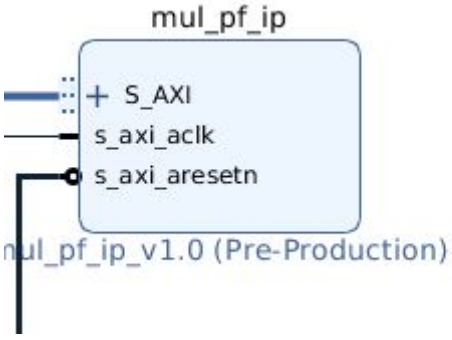
Multiplicador de punto flotante

1. INTRODUCCIÓN.

En el presente trabajo se hablara del desarrollo de un ip core de multiplicación de punto flotante teniendo en cuenta un trabajo anteriormente realizado el cual se referencia en el siguiente link:
https://docs.google.com/document/d/19R4E-d1sGVQ0E300vvSoS4uRPX_amLsE0ZnonkpThBM/edit

Como punto principal se mencionaba el desarrollo de un IP core que se comunica con una unidad de procesamiento (cortex A9) en la placa de desarrollo Artys 7_10. Lo que se tiene es un procesamiento de datos de punto flotante externos al micro desarrollando un hardware específico

Tomando en cuenta el desarrollo de descripción de hardware de la unidad de punto flotante anteriormente echa, se procede al encapsulamiento en ip core, dando como resultante el siguiente bloque



para poder obtener el bloque diseñado propiamente se estableció las pautas de las guías con el ip core de sumador que fue desarrollado en las clases.
Propiamente se realizaron modificaciones las cuales fueron que en el panel de sources el mul_pf_ip_v1_0_S_AXI.vhd, se le inserto el siguiente codigo

```
component Multiplicador_de_punto_flotante is

    port (
        x : in STD_LOGIC_VECTOR (31 downto 0);
        y : in STD_LOGIC_VECTOR (31 downto 0);
        z : out STD_LOGIC_VECTOR (31 downto 0)
    );

end component;
```

asimismo la declaración de la señal que sera usada para conectar la salida del core

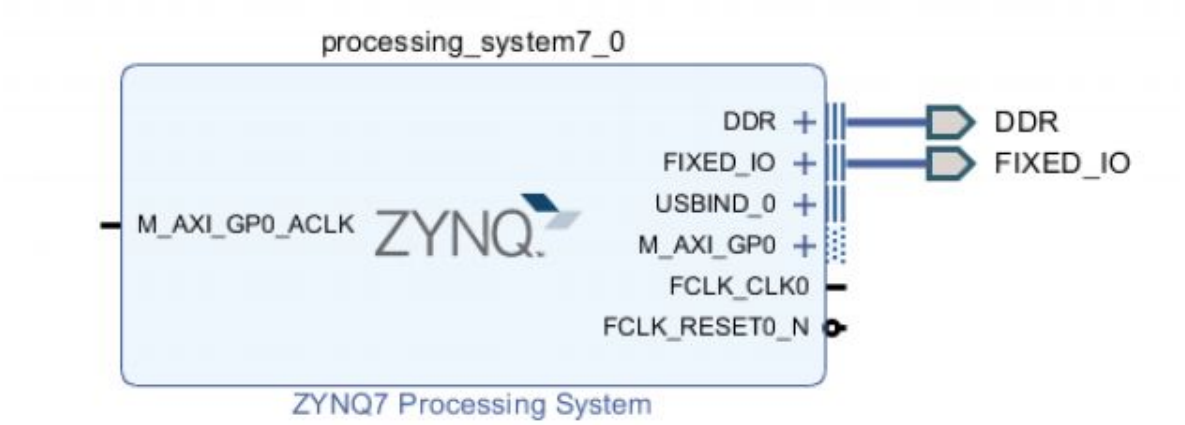
```
signal salMul_pf:std_logic_vector (31 downto 0);
```

para la parte descriptiva de la arquitectura realizar la instanciación del componente propiamente creado

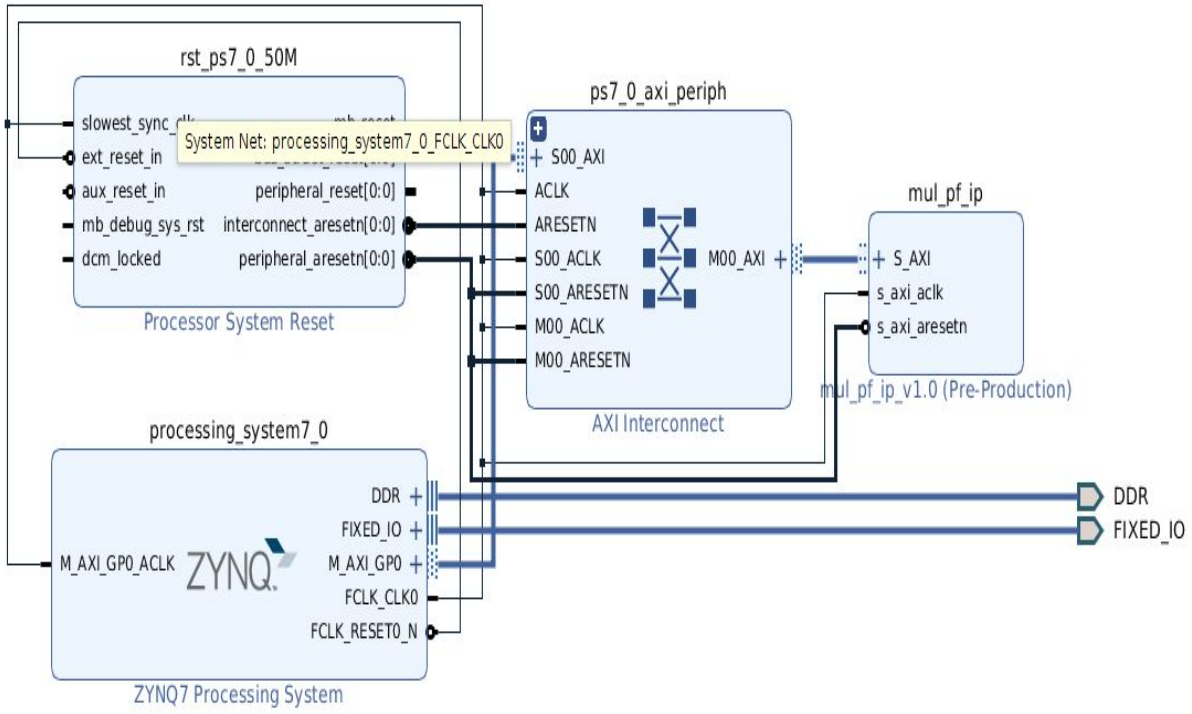
```
-- Add user logic here
inst_mul_pf: Multiplicador_de_punto_flotante
    port map(
        x=>slv_reg0,
        y=>slv_reg1,
        z=>salMul_pf

    );
```

con la instanciación realizada se guarda agreda el proyecto vhd realzado que en este caso es el punto flotante, posteriormente se procede a la sintetización y encapsulamiento

con la creación del encapsulamiento se tiene como consiguiente la conexión con la unidad de procesamiento (cortex A9) para lo cual se tiene el bloque processing system 7_0



Para este se configuró la uart0, se realizó las conexiones con el ip core creado dando como resultado el siguiente diagrama de bloques



con el diagrama ya realizado, se creo el system wrapper.vhd, posteriormente se genera el bitstream y la exportación de hardware.

dejando de lado todo el procedimiento anterior se procede a realizar labor con la SDK. Creando un proyecto multiplicador de punto flotante se tiene el siguiente codigo

```
#include "xparameters.h"
#include "xil_io.h"
#include "mul_pf_ip.h"
#include "xuartps.h"

//================================================


#define UART_DEVICE_ID      XPAR_XUARTPS_0_DEVICE_ID
#define RECV_BUFFER_SIZE    1


float res1;
float opA1=1.1 ;
float opB1=1.5 ;

float res2;
float opA2=2.2 ;
float opB2=3.1 ;

float res3;
float opA3=1.3 ;
float opB3=1.7 ;


XUartPs uart_ps;
XUartPs_Config *config;
```

primeramente se loque se tiene es manejar datos en formato de punto flotante,estas variables declaradas serán operandos definidos que serán enviados al ip core

```
int main (void) {

    u8 data_rec;
    char data;

    //configuracion uart=============================================================
    config=XUartPs_LookupConfig(UART_DEVICE_ID);
    if(NULL==config)
        return XST_FAILURE;

    int status = XUartPs_CfgInitialize(&uart_ps,config,config->BaseAddress);
    if (status!= XST_SUCCESS)
        return XST_FAILURE;

    XUartPs_SetBaudRate(&uart_ps,115200);
    //================================================================================

    while(1)
    {
        data_rec=XUartPs_Recv(&uart_ps,&RecvBuffer,RECV_BUFFER_SIZE);
        if(data_rec!=0)
        {
            data=RecvBuffer;

            xil_printf("operacion %i:" RecvBuffer);
```

para este apartado de código lo que se tiene es la configuración de la uart del micro que comunicara los resultados del producto de punto flotante del ip core por puerto serie.

```
while(1)
{
    data_rec=XUartPs_Recv(&uart_ps,&RecvBuffer,RECV_BUFFER_SIZE);
    if(data_rec!=0)
    {
        data=RecvBuffer;

        xil_printf("operacion %i:",RecvBuffer);

    }


    switch(data){
    case 49:
            xil_printf("primera multiplcacion: \r\n");

            MUL_PF_IP_mWriteReg(XPAR_MUL_PF_IP_S_AXI_BASEADDR, MUL_PF_IP_S_AXI_SLV_R
            MUL_PF_IP_mWriteReg(XPAR_MUL_PF_IP_S_AXI_BASEADDR, MUL_PF_IP_S_AXI_SLV_R
            res1 = (float)MUL_PF_IP_mReadReg(XPAR_MUL_PF_IP_S_AXI_BASEADDR, MUL_PF_I

             xil_printf("multiplicacion: %f * %f = %f\r\n", opA1, opB1, res1);

        break;
    case 50:
        xil_printf("segunda multiplcacion: \r\n");

        MUL_PF_IP_mWriteReg(XPAR_MUL_PF_IP_S_AXI_BASEADDR, MUL_PF_IP_S_AXI_SLV_REG0
```

para este apartado es mas para recepción de los datos del registro que escribe al ip core y resultantes que arroja por la operación establecida

```
            MUL_PF_IP_mWriteReg(XPAR_MUL_PF_IP_S_AXI_BASEADDR, MUL_PF_IP_S_AXI_SLV_R
            MUL_PF_IP_mWriteReg(XPAR_MUL_PF_IP_S_AXI_BASEADDR, MUL_PF_IP_S_AXI_SLV_R
            res1 = (float)MUL_PF_IP_mReadReg(XPAR_MUL_PF_IP_S_AXI_BASEADDR, MUL_PF_I

             xil_printf("multiplicacion: %f * %f = %f\r\n", opA1, opB1, res1);

        break;
    case 50:
        xil_printf("segunda multiplcacion: \r\n");

        MUL_PF_IP_mWriteReg(XPAR_MUL_PF_IP_S_AXI_BASEADDR, MUL_PF_IP_S_AXI_SLV_REG0
        MUL_PF_IP_mWriteReg(XPAR_MUL_PF_IP_S_AXI_BASEADDR, MUL_PF_IP_S_AXI_SLV_REG1
        res2 = (float) MUL_PF_IP_mReadReg(XPAR_MUL_PF_IP_S_AXI_BASEADDR, MUL_PF_IP_S

         xil_printf("multiplicacion: %f * %f = %f\r\n", opA2, opB2, res2);

        break;

    case 51:
        xil_printf("tercera multiplcacion: \r\n");

        MUL_PF_IP_mWriteReg(XPAR_MUL_PF_IP_S_AXI_BASEADDR, MUL_PF_IP_S_AXI_SLV_REG0
        MUL_PF_IP_mWriteReg(XPAR_MUL_PF_IP_S_AXI_BASEADDR, MUL_PF_IP_S_AXI_SLV_REG1
        res3 = (float)MUL_PF_IP_mReadReg(XPAR_MUL_PF_IP_S_AXI_BASEADDR, MUL_PF_IP_S_

        xil_printf("multiplicacion: %d * %d = %d\r\n", opA3, opB3, res3);
```

con está ultima imagen se muestra las diferentes resultantes que se obtiene por la instrucciones recibidas por uart