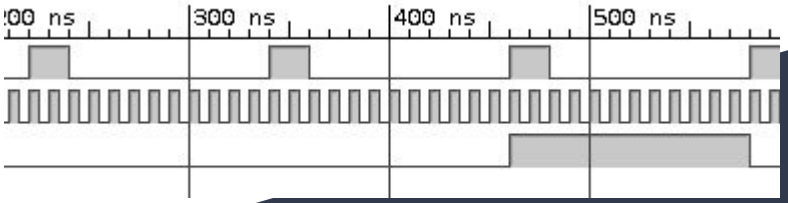
Detector de períodos de un tren de pulsos con FPGA (IP CORE & CORTEX A9)



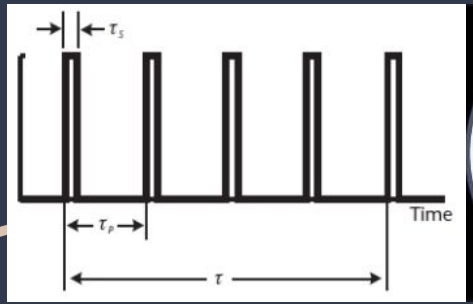
Autor: Ing. Fabián Ezequiel Urdaniz
Docente: Ing. Nicolás Álvarez
CESE -Microarquitecturas y Softcores

FACULTAD DE INGENIERIA
Universidad de Buenos Aires

LSE Laboratorio de Sistemas Embebidos

Descripción de trabajo

Este mini proyecto busca implementar con IP CORE y CORTEX A9 la detección de un tren de pulsos emitidos por un radar y lograr determinar el periodo de estos de manera sincrónica.



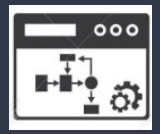
Introducción

El trabajo es la continuación del proyecto final de la materia Circuitos lógicos programables y cómo se habíamos expresado no tiene relación directa con el proyecto de finalización de carrera pero se desprende de un proyecto mayor que tiene como objetivo el control de señales de un radar con FPGA.

Objetivo

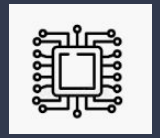
Desarrollar un bloque de hardware digital en lenguaje VHDL (IP Core) que integre un sistema base de procesamiento junto con el micro Cortex A9. El funcionamiento del mismo será implementado a través de un código C. El sistema debe ser capaz de detectar el período de una señal del tipo tren de pulsos y emitir una señal de detección cuya duración es el período calculado. En caso de fallas o errores emitir una señal de falla. Estos eventos deben ser informados en los correspondientes registros.

Elementos de trabajo anteriores

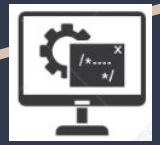


Máquinas de estados

Conversiones
Contadores



Combinacionales



Vivado - VHDL

La implementación se basó en los temas desarrollados en la materia.

El sistema está compuesto principalmente por un componente detector de señal con

Entradas:

Clock (reloj interno).

Señal entrada (secuencia).

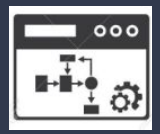
Salidas:

Señal de detección.

Señal de falla.

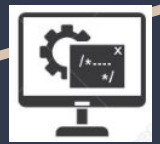
Elementos de trabajo agregados

Registros 32bits



IP CORE DETECTOR

ZYNQ - Cortex-A9



Vivado - SDK

La implementación se basó en los ejemplos desarrollados en la materia.

El sistema está compuesto principalmente por un IP CORE detector y un sistema de procesamiento ZYNQ7

Lenguaje:

VHDL

C

Salidas:

Detect : Logic_vector (31 downto 0)

Error : Logic_vector (31 downto 0)

Nuevo IP Detector

Detector_ip_v1

```
entity detector_ip_v1_0 is
    generic (
        C_S_AXI_DATA_WIDTH  : integer    := 32;
        C_S_AXI_ADDR_WIDTH  : integer    := 4
    );
    port (
        -- Users to add ports here
        ------------------------------
        sig_secuencia_i : in std_logic;
        clk_i           : in std_logic;
        sig_detec_o     : out std_logic;
        fail_o          : out std_logic;
```

```
architecture arch_imp of detector_ip_v1_0 is
    -- component declaration
    component detector_ip_v1_0_S_AXI is
        generic (
        C_S_AXI_DATA_WIDTH  : integer    := 32;
        C_S_AXI_ADDR_WIDTH  : integer    := 4
        );
        port (
        ----------------------------
        SIG_SECUENCIA_i : in std_logic;
        CLK_i           : in std_logic;
        SIG_DETEC_o     : out std_logic;
        FAIL_o          : out std_logic;
```

```
-- Instantiation of Axi Bus Interface S_AXI
detector_ip_v1_0_S_AXI_inst : detector_ip_v1_0_S_AXI
    generic map (
        C_S_AXI_DATA_WIDTH  => C_S_AXI_DATA_WIDTH,
        C_S_AXI_ADDR_WIDTH  => C_S_AXI_ADDR_WIDTH
    )
    port map (
        ----------------------
        SIG_SECUENCIA_i => sig_secuencia_i,
        CLK_i=> clk_i,
        SIG_DETEC_o=> sig_detec_o,
        FAIL_o => fail_o,
        ----------------------
```

Nuevo IP Detector

Detector_ip_v1_S_AXI

```
entity detector_ip_v1_0_S_AXI is
    generic (
        C_S_AXI_DATA_WIDTH  : integer   := 32;
        -- Width of S_AXI address bus
        C_S_AXI_ADDR_WIDTH  : integer   := 4
    );
    port (
        -- Users to add ports here
        ----------------------------------------
        SIG_SECUENCIA_i : in std_logic;
        CLK_i           : in std_logic;
        SIG_DETEC_o     : out std_logic;
        FAIL_o          : out std_logic;
```

```
---------------- signal -----------------
signal slv_reg0_detc: std_logic_vector(31 downto 0);
signal slv_reg1_err: std_logic_vector(31 downto 0);

--------------- componente detector ---------------------
component detector is
    port(
       ---------- i/o -------
        sig_secuencia_i : in std_logic;            -- signal in.
        clk_i           : in std_logic;            -- clock.
        sig_detec_o     : out std_logic;           -- detec sign
        fail_o          : out std_logic;           -- fail.
       ---------- Registros -------
        detc_o          : out std_logic_vector(31 downto 0);
        err_o           : out std_logic_vector(31 downto 0)
        );
end component;
```

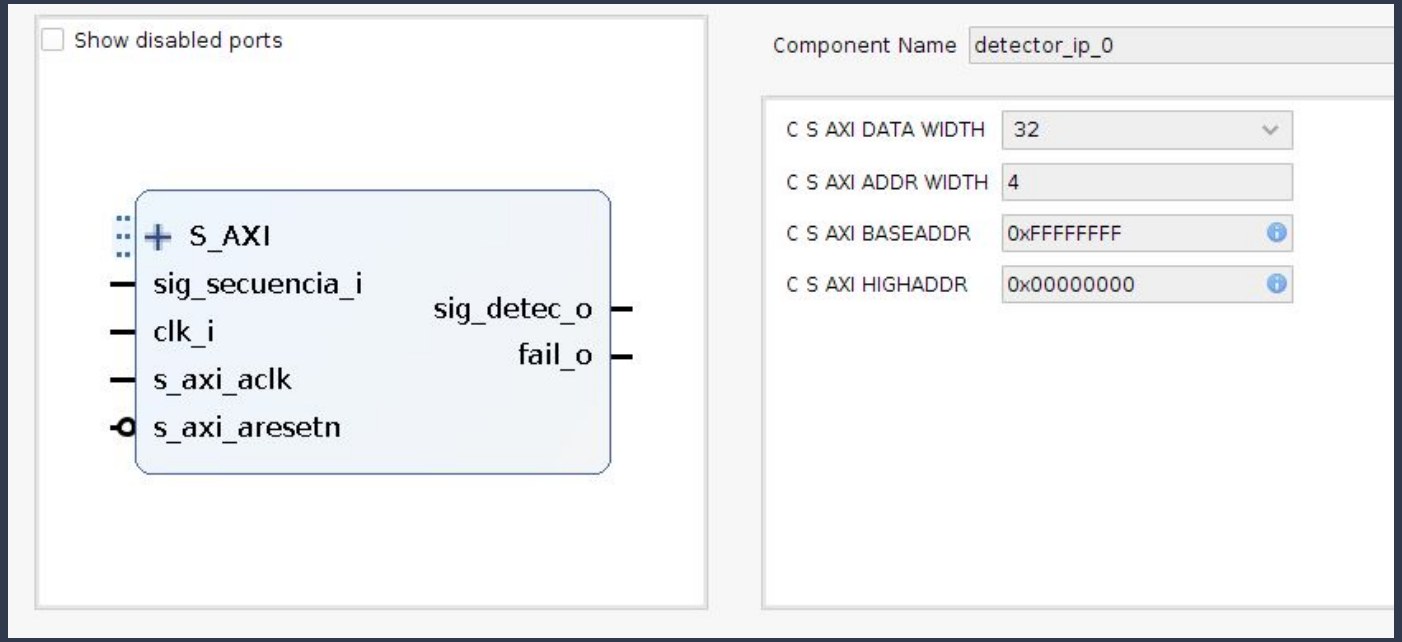
```
-- Add user logic here
-----------------------------------
inst_detec:detector
port map (
       ------- i/o ---------
   sig_secuencia_i  => SIG_SECUENCIA_i,
   clk_i => CLK_i,
   sig_detec_o  => SIG_DETEC_o,
   fail_o  => FAIL_o,

        ---------- Registros -------
    detc_o => slv_reg0_detc,
    err_o  => slv_reg1_err
   );
-------------------------------------
-- User logic ends
```

Nuevo IP Detector

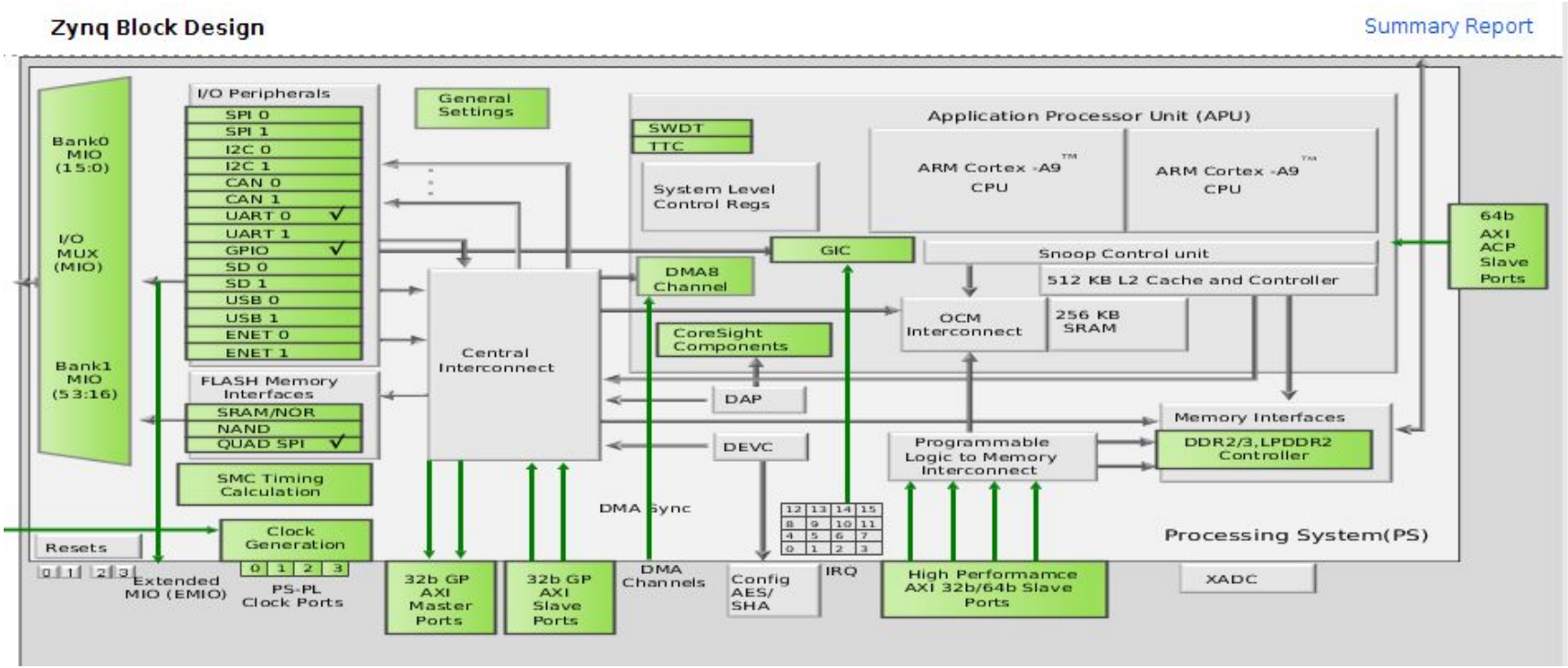
Detector_ip_v1_S_AXI

```
process (slv_reg0_detc, slv_reg1_err, slv_reg2, slv_reg3, axi_araddr, S_AXI_ARESETN, slv_reg_rden)
variable loc_addr :std_logic_vector(OPT_MEM_ADDR_BITS downto 0);
begin
    -- Address decoding for reading registers
    loc_addr := axi_araddr(ADDR_LSB + OPT_MEM_ADDR_BITS downto ADDR_LSB);
    case loc_addr is
      when b"00" =>
        reg_data_out <= slv_reg0_detc;
      when b"01" =>
        reg_data_out <= slv_reg1_err;
      when b"10" =>
        reg_data_out <= slv_reg2;
      when b"11" =>
        reg_data_out <= slv_reg3;
      when others =>
        reg_data_out  <= (others => '0');
    end case;
end process;
```

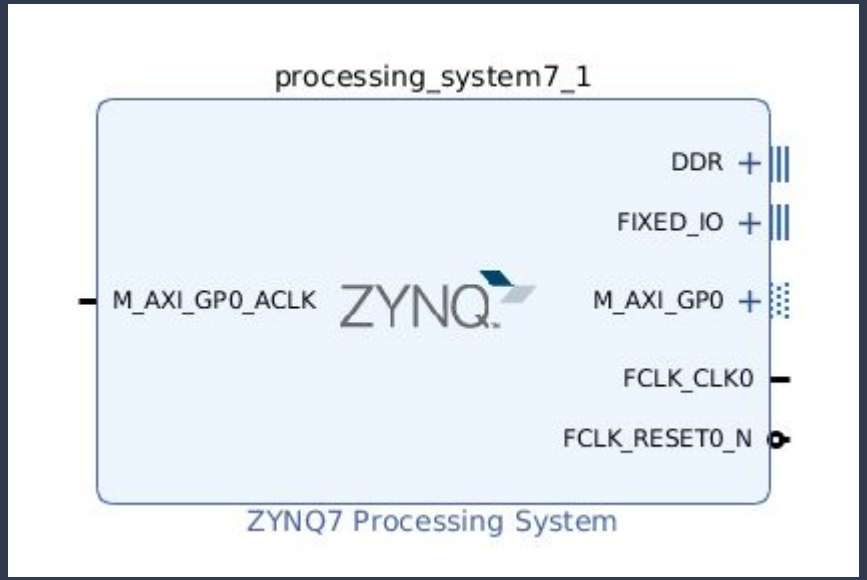
Detector IP – Propiedades



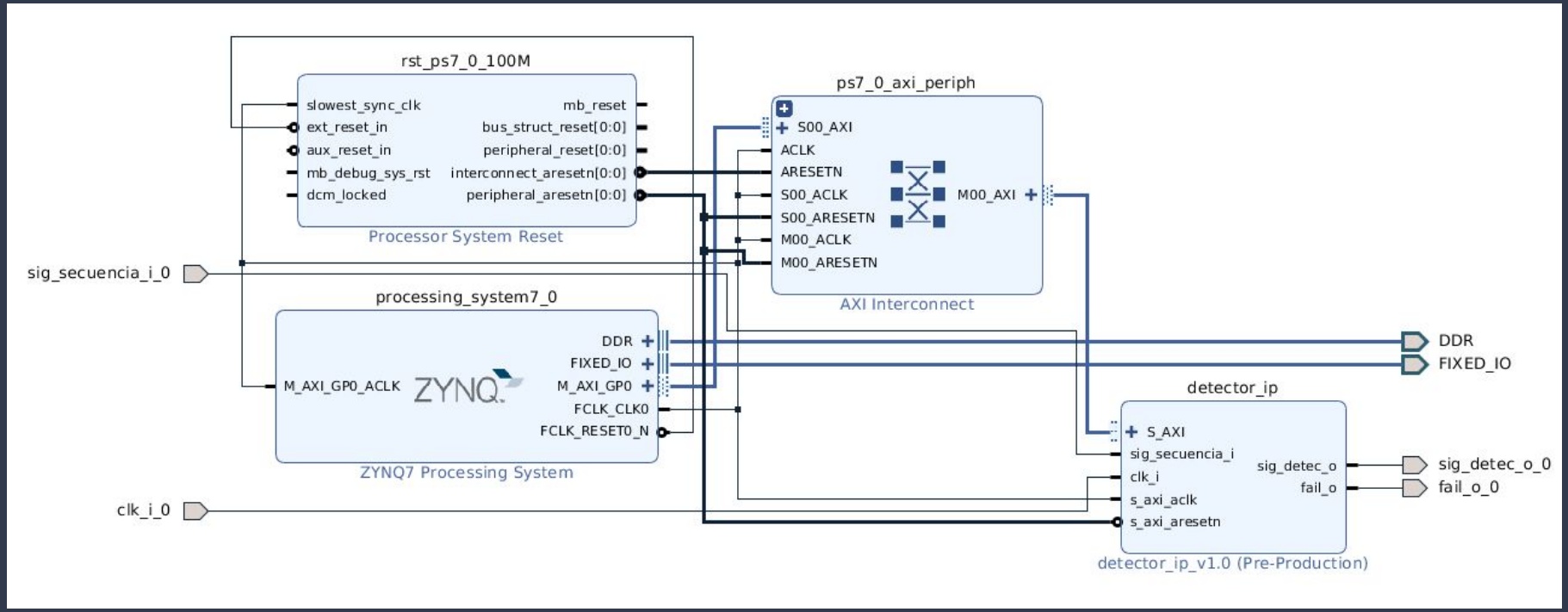
Sistema procesador - ZYNQ -A9



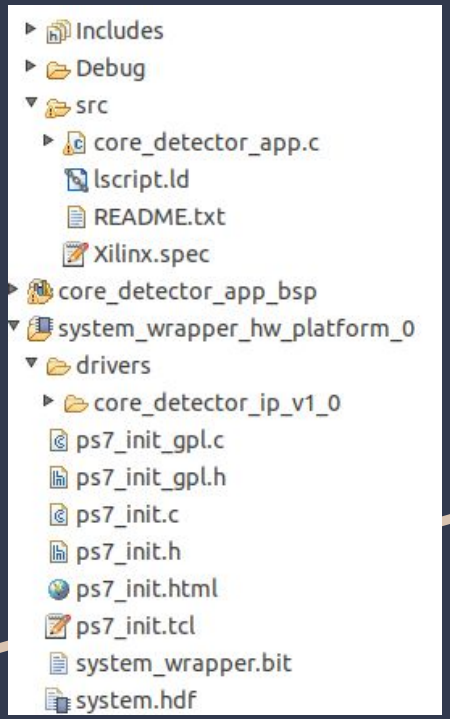
ZYNQ - Propiedades



Sistema integrado



Xilinx SDK



System wrapper

Design Information

Target FPGA Device:	7z010
Part:	xc7z010clg400-1
Created With:	Vivado 2018.1
Created On:	Mon Aug 24 15:05:37 2020

Address Map for processor ps7_cortexa9_[0-1]

Cell	Base Addr	High Addr	Slave I/f	M
core_detector_ip	0x43c00000	0x43c0ffff	S_AXI	R
ps7_intc_dist_0	0xf8f01000	0xf8f01fff		R

Peripheral Drivers

Drivers present in the Board Support Package.

core_detector_ip core_detector_ip
ps7_afi_0 generic
ps7_afi_1 generic
ps7_afi_2 generic

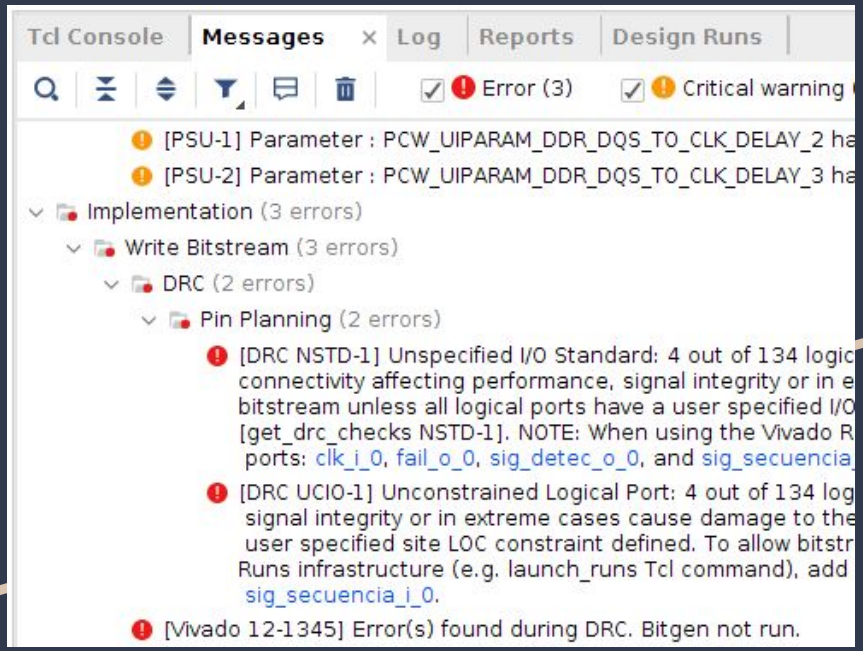
Código - Aplicación Detector

```
#ifndef CORE_DETECTOR_IP_H
#define CORE_DETECTOR_IP_H

#include "xil_types.h"
#include "xstatus.h"

#define CORE_DETECTOR_IP_S_AXI_SLV_REG0_OFFSET 0
#define CORE_DETECTOR_IP_S_AXI_SLV_REG1_OFFSET 4
#define CORE_DETECTOR_IP_S_AXI_SLV_REG2_OFFSET 8
#define CORE_DETECTOR_IP_S_AXI_SLV_REG3_OFFSET 12



#define CORE_DETECTOR_IP_mWriteReg(BaseAddress, RegOffset, Data) \
    Xil_Out32((BaseAddress) + (RegOffset), (u32)(Data))

#define CORE_DETECTOR_IP_mReadReg(BaseAddress, RegOffset) \
    Xil_In32((BaseAddress) + (RegOffset))


XStatus CORE_DETECTOR_IP_Reg_SelfTest(void * baseaddr_p);

#endif
```

Código - Aplicación Detector

```
#include "xparameters.h"
#include "xil_io.h"
#include "core_detector_ip.h"

int main (void) {

    //--- Variables --
    int detec = 99;
    int fail = 99;

    //-- Inicializacion de programa ---

    xil_printf("-- Detector de tren de pulsos con IP detector propio --\r\n");

    while (1){

    //-- Read reg0 -- detec
    detec = CORE_DETECTOR_IP_mReadReg(XPAR_CORE_DETECTOR_IP_S_AXI_BASEADDR,CORE_DETECTOR_IP_S_AXI_SLV_REG0_OFFSET);

    //-- Read reg1 -- fail
    fail = CORE_DETECTOR_IP_mReadReg(XPAR_CORE_DETECTOR_IP_S_AXI_BASEADDR,CORE_DETECTOR_IP_S_AXI_SLV_REG1_OFFSET);

    xil_printf("detector tren de pulso: %d\r\n", detec);
    xil_printf("falla en deteccion: %d\r\n", fail);

    sleep(1);
    }

}
```

FALLAS en BITSTREAM



RUN SYNTHESIS

RUN IMPLEMENT

Errores en la generación de BITSTREAM

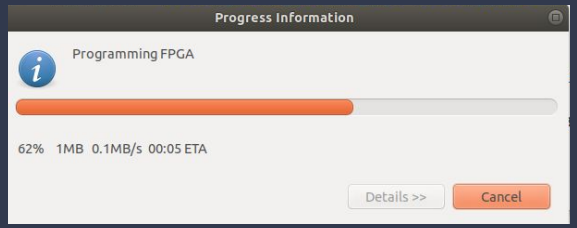
SOLUCIÓN en BITSTREAM

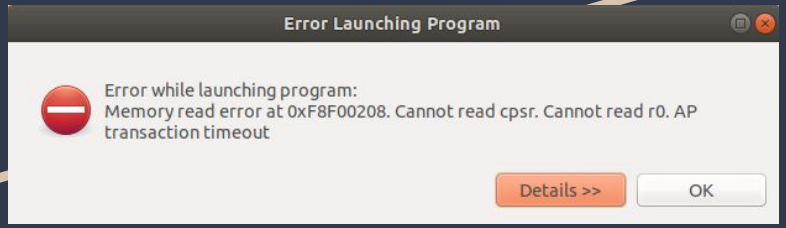
RUN SYNTHESIS IMPLEMENT

ame	Constraints	Status
synth_1 (active)	**constrs_1**	**synth_design Cor**
impl_1	constrs_1	write_bitstream Con
Out-of-Context Module Runs		
system		Submodule Runs Cc
system_processing_system7_0_0_synt...	system_pr...	synth_design Comp
system_core_detector_ip_0_0_synth_1	system_c...	synth_design Comp
system_rst_ps7_0_100M_0_synth_1	system_rs...	synth_design Comp
system_auto_pc_0_synth_1	system_a...	synth_design Comp

Generación de BITSTREAM

FALLAS en programación remota

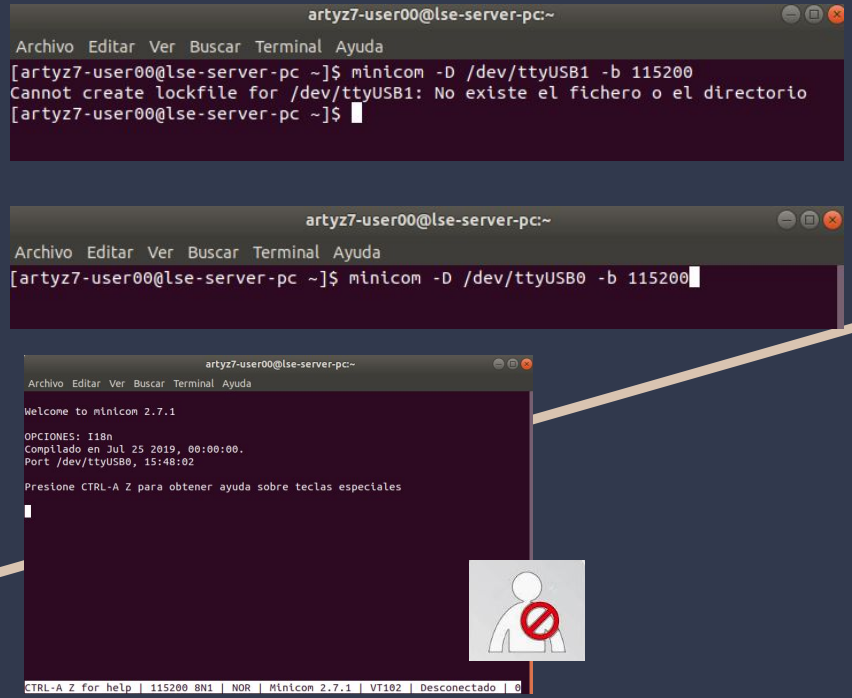




Conexión remota

Errores en la programación

FALLAS en conexión kit FPGA



Conexión remota ttyUSB0

Errores conexión ttyUSB1

Conclusiones

Se completaron todas las etapas del proyecto desde la creación y adaptación propia de un IP CORE propia hasta la interacción con el procesador CORTEX A9 del ZYNQ. Si bien la funcionalidad del sistema completo depende de las entradas y salidas externas de la placa. Los registros de eventos son detectados e impresos por la terminal.

Se pudoo generar BITSTREAM.
No se puede acceder al KIT FPGA.

Los resultados obtenidos son los esperados.

¿ Preguntas ?

Fin

¡¡Muchas gracias por su atención!!

Consultas: urdanizezequiel@gmail.com