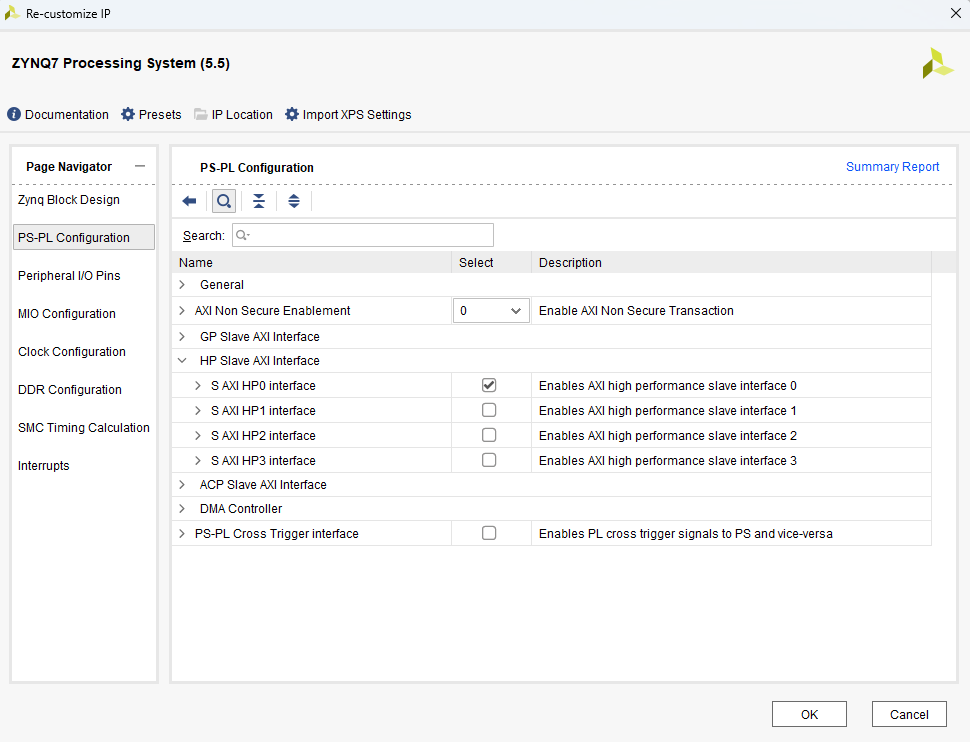
Creacion de AXI Stream

Implementación del ZYNQ:



Tabla

Descripción generada automáticamente con confianza baja

Clock Wizard

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

XADC

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Xconstant para el address

Interfaz de usuario gráfica

Descripción generada automáticamente

AXI DMA:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity ADC\_Stream\_v1\_0\_M00\_AXIS is

generic (

-- Users to add parameters here

-- User parameters ends

-- Do not modify the parameters beyond this line

-- Width of S\_AXIS address bus. The slave accepts the read and write addresses of width C\_M\_AXIS\_TDATA\_WIDTH.

C\_M\_AXIS\_TDATA\_WIDTH : integer := 32;

-- Start count is the number of clock cycles the master will wait before initiating/issuing any transaction.

C\_M\_START\_COUNT : integer := 32

);

port (

-- Users to add ports here

-- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

-- Codigo agregado por Fabian

-- Se agregan los puertos para el IP

data\_in : in std\_logic\_vector(15 downto 0);

clk\_adc : in std\_logic;

-- %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

-- User ports ends

-- Do not modify the ports beyond this line

-- Global ports

M\_AXIS\_ACLK : in std\_logic;

--

M\_AXIS\_ARESETN : in std\_logic;

-- Master Stream Ports. TVALID indicates that the master is driving a valid transfer, A transfer takes place when both TVALID and TREADY are asserted.

M\_AXIS\_TVALID : out std\_logic;

-- TDATA is the primary payload that is used to provide the data that is passing across the interface from the master.

M\_AXIS\_TDATA : out std\_logic\_vector(C\_M\_AXIS\_TDATA\_WIDTH-1 downto 0);

-- TSTRB is the byte qualifier that indicates whether the content of the associated byte of TDATA is processed as a data byte or a position byte.

M\_AXIS\_TSTRB : out std\_logic\_vector((C\_M\_AXIS\_TDATA\_WIDTH/8)-1 downto 0);

-- TLAST indicates the boundary of a packet.

M\_AXIS\_TLAST : out std\_logic;

-- TREADY indicates that the slave can accept a transfer in the current cycle.

M\_AXIS\_TREADY : in std\_logic

);

end ADC\_Stream\_v1\_0\_M00\_AXIS;

architecture implementation of ADC\_Stream\_v1\_0\_M00\_AXIS is

-- Total number of output data

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

-- Codigo agregado por Fabian

-- Este valor se edita para variar el tamaño del buffer de salida

constant NUMBER\_OF\_OUTPUT\_WORDS : integer := 8191; --13 bits --8;

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

-- function called clogb2 that returns an integer which has the

-- value of the ceiling of the log base 2.

function clogb2 (bit\_depth : integer) return integer is

variable depth : integer := bit\_depth;

variable count : integer := 1;

begin

for clogb2 in 1 to bit\_depth loop -- Works for up to 32 bit integers

if (bit\_depth <= 2) then

count := 1;

else

if(depth <= 1) then

count := count;

else

depth := depth / 2;

count := count + 1;

end if;

end if;

end loop;

return(count);

end;

-- WAIT\_COUNT\_BITS is the width of the wait counter.

constant WAIT\_COUNT\_BITS : integer := clogb2(C\_M\_START\_COUNT-1);

-- In this example, Depth of FIFO is determined by the greater of

-- the number of input words and output words.

constant depth : integer := NUMBER\_OF\_OUTPUT\_WORDS;

-- bit\_num gives the minimum number of bits needed to address 'depth' size of FIFO

constant bit\_num : integer := clogb2(depth);

-- Define the states of state machine

-- The control state machine oversees the writing of input streaming data to the FIFO,

-- and outputs the streaming data from the FIFO

type state is ( IDLE, -- This is the initial/idle state

INIT\_COUNTER, -- This state initializes the counter, once

-- the counter reaches C\_M\_START\_COUNT count,

-- the state machine changes state to SEND\_STREAM

SEND\_STREAM); -- In this state the

-- stream data is output through M\_AXIS\_TDATA

-- State variable

signal mst\_exec\_state : state;

-- Example design FIFO read pointer

signal read\_pointer : integer range 0 to depth-1;

-- AXI Stream internal signals

--wait counter. The master waits for the user defined number of clock cycles before initiating a transfer.

signal count : std\_logic\_vector(WAIT\_COUNT\_BITS-1 downto 0);

--streaming data valid

signal axis\_tvalid : std\_logic;

--streaming data valid delayed by one clock cycle

signal axis\_tvalid\_delay : std\_logic;

--Last of the streaming data

signal axis\_tlast : std\_logic;

--Last of the streaming data delayed by one clock cycle

signal axis\_tlast\_delay : std\_logic;

--FIFO implementation signals

signal stream\_data\_out : std\_logic\_vector(C\_M\_AXIS\_TDATA\_WIDTH-1 downto 0);

signal tx\_en : std\_logic;

--The master has issued all the streaming data stored in FIFO

signal tx\_done : std\_logic;

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

-- Codigo agregado por Fabian

-- Senales agregadas para funciones del fifo

type datos\_fifo is array (0 to NUMBER\_OF\_OUTPUT\_WORDS + 5 ) of std\_logic\_vector(15 downto 0);

signal dfifo : datos\_fifo := (others => (others => '0'));

signal act : std\_logic;

signal contadorfifo : integer := 0;

signal activar : std\_logic := '0';

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

begin

-- I/O Connections assignments

M\_AXIS\_TVALID <= axis\_tvalid\_delay;

M\_AXIS\_TDATA <= stream\_data\_out;

M\_AXIS\_TLAST <= axis\_tlast\_delay;

M\_AXIS\_TSTRB <= (others => '1');

-- Control state machine implementation

process(M\_AXIS\_ACLK)

begin

if (rising\_edge (M\_AXIS\_ACLK)) then

if(M\_AXIS\_ARESETN = '0') then

-- Synchronous reset (active low)

mst\_exec\_state <= IDLE;

count <= (others => '0');

else

case (mst\_exec\_state) is

when IDLE =>

-- The slave starts accepting tdata when

-- there tvalid is asserted to mark the

-- presence of valid streaming data

--if (count = "0")then

mst\_exec\_state <= INIT\_COUNTER;

--else

-- mst\_exec\_state <= IDLE;

--end if;

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

-- Codigo agregado por Fabian

-- Inicializacion de contadores

contadorfifo <= 0;

read\_pointer <= 0;

activar <= '0';

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

when INIT\_COUNTER =>

-- This state is responsible to wait for user defined C\_M\_START\_COUNT

-- number of clock cycles.

-- if ( count = std\_logic\_vector(to\_unsigned((C\_M\_START\_COUNT - 1), WAIT\_COUNT\_BITS))) then

-- mst\_exec\_state <= SEND\_STREAM;

-- else

-- count <= std\_logic\_vector (unsigned(count) + 1);

-- mst\_exec\_state <= INIT\_COUNTER;

-- end if;

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

-- Codigo agregado por Fabian

-- Pasa directo al llenado del Stream

mst\_exec\_state <= SEND\_STREAM;

act <= '0';

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

when SEND\_STREAM =>

-- The example design streaming master functionality starts

-- when the master drives output tdata from the FIFO and the slave

-- has finished storing the S\_AXIS\_TDATA

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

-- Codigo agregado por Fabian

-- LLenado del buffer intermedio

if activar = '0' and clk\_adc = '1' then

activar <= '1';

if contadorfifo < NUMBER\_OF\_OUTPUT\_WORDS+5 then

dfifo(contadorfifo) <= data\_in;

contadorfifo <= contadorfifo + 1;

end if;

elsif clk\_adc = '0' then

activar <= '0';

end if;

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

-- Codigo agregado por Fabian

-- Envio de datos por el AXI Stream

axis\_tvalid <= '1';

if read\_pointer <= NUMBER\_OF\_OUTPUT\_WORDS and M\_AXIS\_TREADY = '1' then

read\_pointer <= read\_pointer + 1;

stream\_data\_out <= "0000000000000000" & dfifo(read\_pointer);

act <= '1';

elsif read\_pointer > NUMBER\_OF\_OUTPUT\_WORDS then

mst\_exec\_state <= IDLE;

elsif M\_AXIS\_TREADY = '0' and act = '1' then

mst\_exec\_state <= IDLE;

end if;

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

-- if (tx\_done = '1') then

-- mst\_exec\_state <= IDLE;

-- else

-- mst\_exec\_state <= SEND\_STREAM;

-- end if;

when others =>

mst\_exec\_state <= IDLE;

end case;

end if;

end if;

end process;

--tvalid generation

--axis\_tvalid is asserted when the control state machine's state is SEND\_STREAM and

--number of output streaming data is less than the NUMBER\_OF\_OUTPUT\_WORDS.

-- axis\_tvalid <= '1' when ((mst\_exec\_state = SEND\_STREAM) and (read\_pointer < NUMBER\_OF\_OUTPUT\_WORDS)) else '0';

-- AXI tlast generation

-- axis\_tlast is asserted number of output streaming data is NUMBER\_OF\_OUTPUT\_WORDS-1

-- (0 to NUMBER\_OF\_OUTPUT\_WORDS-1)

axis\_tlast <= '1' when (read\_pointer = NUMBER\_OF\_OUTPUT\_WORDS-1) else '0';

-- Delay the axis\_tvalid and axis\_tlast signal by one clock cycle

-- to match the latency of M\_AXIS\_TDATA

process(M\_AXIS\_ACLK)

begin

if (rising\_edge (M\_AXIS\_ACLK)) then

if(M\_AXIS\_ARESETN = '0') then

axis\_tvalid\_delay <= '0';

axis\_tlast\_delay <= '0';

else

axis\_tvalid\_delay <= axis\_tvalid;

axis\_tlast\_delay <= axis\_tlast;

end if;

end if;

end process;

--read\_pointer pointer

-- process(M\_AXIS\_ACLK)

-- begin

-- if (rising\_edge (M\_AXIS\_ACLK)) then

-- if(M\_AXIS\_ARESETN = '0') then

-- read\_pointer <= 0;

-- tx\_done <= '0';

-- else

-- if (read\_pointer <= NUMBER\_OF\_OUTPUT\_WORDS-1) then

-- if (tx\_en = '1') then

-- -- read pointer is incremented after every read from the FIFO

-- -- when FIFO read signal is enabled.

-- read\_pointer <= read\_pointer + 1;

-- tx\_done <= '0';

-- end if;

-- elsif (read\_pointer = NUMBER\_OF\_OUTPUT\_WORDS) then

-- -- tx\_done is asserted when NUMBER\_OF\_OUTPUT\_WORDS numbers of streaming data

-- -- has been out.

-- tx\_done <= '1';

-- end if;

-- end if;

-- end if;

-- end process;

--FIFO read enable generation

-- tx\_en <= M\_AXIS\_TREADY and axis\_tvalid;

-- FIFO Implementation

-- Streaming output data is read from FIFO

-- process(M\_AXIS\_ACLK)

-- variable sig\_one : integer := 1;

-- begin

-- if (rising\_edge (M\_AXIS\_ACLK)) then

-- if(M\_AXIS\_ARESETN = '0') then

-- stream\_data\_out <= std\_logic\_vector(to\_unsigned(sig\_one,C\_M\_AXIS\_TDATA\_WIDTH));

-- elsif (tx\_en = '1') then -- && M\_AXIS\_TSTRB(byte\_index)

-- stream\_data\_out <= std\_logic\_vector( to\_unsigned(read\_pointer,C\_M\_AXIS\_TDATA\_WIDTH) + to\_unsigned(sig\_one,C\_M\_AXIS\_TDATA\_WIDTH));

-- end if;

-- end if;

-- end process;

-- Add user logic here

-- User logic ends

end implementation;

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity ADC\_Stream\_v1\_0 is

generic (

-- Users to add parameters here

-- User parameters ends

-- Do not modify the parameters beyond this line

-- Parameters of Axi Master Bus Interface M00\_AXIS

C\_M00\_AXIS\_TDATA\_WIDTH : integer := 32;

C\_M00\_AXIS\_START\_COUNT : integer := 32

);

port (

-- Users to add ports here

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

-- Codigo agregado por Fabian

-- Se agregan los puertos necesarios para el funcionamiento del IP

data\_in : in std\_logic\_vector(15 downto 0);

clk\_adc : in std\_logic;

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

-- User ports ends

-- Do not modify the ports beyond this line

-- Ports of Axi Master Bus Interface M00\_AXIS

m00\_axis\_aclk : in std\_logic;

m00\_axis\_aresetn : in std\_logic;

m00\_axis\_tvalid : out std\_logic;

m00\_axis\_tdata : out std\_logic\_vector(C\_M00\_AXIS\_TDATA\_WIDTH-1 downto 0);

m00\_axis\_tstrb : out std\_logic\_vector((C\_M00\_AXIS\_TDATA\_WIDTH/8)-1 downto 0);

m00\_axis\_tlast : out std\_logic;

m00\_axis\_tready : in std\_logic

);

end ADC\_Stream\_v1\_0;

architecture arch\_imp of ADC\_Stream\_v1\_0 is

-- component declaration

component ADC\_Stream\_v1\_0\_M00\_AXIS is

generic (

C\_M\_AXIS\_TDATA\_WIDTH : integer := 32;

C\_M\_START\_COUNT : integer := 32

);

port (

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

data\_in : in std\_logic\_vector(15 downto 0);

clk\_adc : in std\_logic;

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_AXIS\_ACLK : in std\_logic;

M\_AXIS\_ARESETN : in std\_logic;

M\_AXIS\_TVALID : out std\_logic;

M\_AXIS\_TDATA : out std\_logic\_vector(C\_M\_AXIS\_TDATA\_WIDTH-1 downto 0);

M\_AXIS\_TSTRB : out std\_logic\_vector((C\_M\_AXIS\_TDATA\_WIDTH/8)-1 downto 0);

M\_AXIS\_TLAST : out std\_logic;

M\_AXIS\_TREADY : in std\_logic

);

end component ADC\_Stream\_v1\_0\_M00\_AXIS;

begin

-- Instantiation of Axi Bus Interface M00\_AXIS

ADC\_Stream\_v1\_0\_M00\_AXIS\_inst : ADC\_Stream\_v1\_0\_M00\_AXIS

generic map (

C\_M\_AXIS\_TDATA\_WIDTH => C\_M00\_AXIS\_TDATA\_WIDTH,

C\_M\_START\_COUNT => C\_M00\_AXIS\_START\_COUNT

)

port map (

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

data\_in => data\_in,

clk\_adc => clk\_adc,

--%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

M\_AXIS\_ACLK => m00\_axis\_aclk,

M\_AXIS\_ARESETN => m00\_axis\_aresetn,

M\_AXIS\_TVALID => m00\_axis\_tvalid,

M\_AXIS\_TDATA => m00\_axis\_tdata,

M\_AXIS\_TSTRB => m00\_axis\_tstrb,

M\_AXIS\_TLAST => m00\_axis\_tlast,

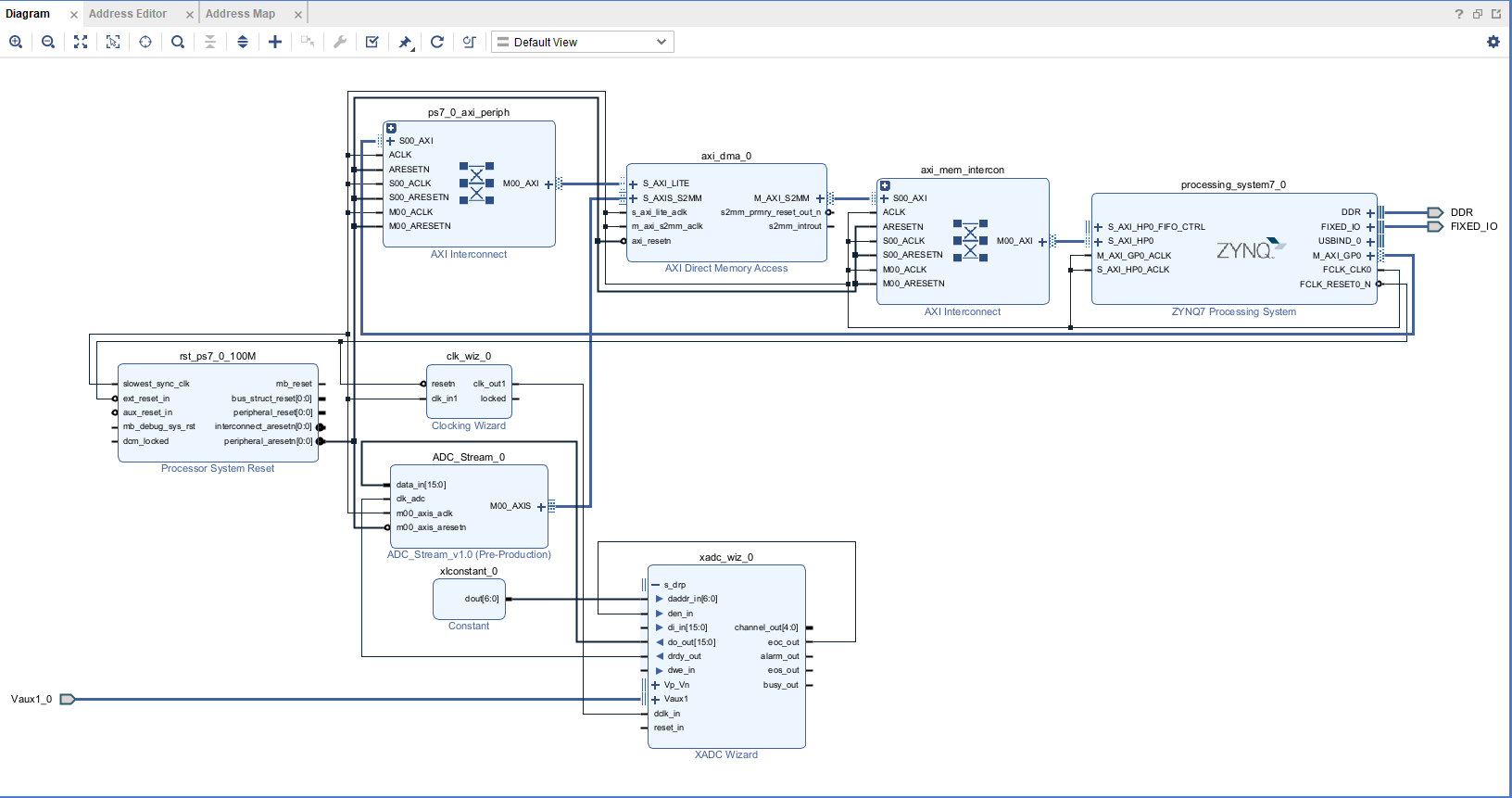
M\_AXIS\_TREADY => m00\_axis\_tready

);

-- Add user logic here

-- User logic ends

end arch\_imp;



## This file is a general .xdc for the PYNQ-Z1 board Rev. C

## To use it in a project:

## - uncomment the lines corresponding to used pins

## - rename the used ports (in each line, after get\_ports) according to the top level signal names in the project

## Clock signal 125 MHz

#set\_property -dict { PACKAGE\_PIN H16 IOSTANDARD LVCMOS33 } [get\_ports { sysclk }]; #IO\_L13P\_T2\_MRCC\_35 Sch=sysclk

#create\_clock -add -name sys\_clk\_pin -period 8.00 -waveform {0 4} [get\_ports { sysclk }];

##Switches

#set\_property -dict { PACKAGE\_PIN M20 IOSTANDARD LVCMOS33 } [get\_ports { sw[0] }]; #IO\_L7N\_T1\_AD2N\_35 Sch=sw[0]

#set\_property -dict { PACKAGE\_PIN M19 IOSTANDARD LVCMOS33 } [get\_ports { sw[1] }]; #IO\_L7P\_T1\_AD2P\_35 Sch=sw[1]

##RGB LEDs

#set\_property -dict { PACKAGE\_PIN L15 IOSTANDARD LVCMOS33 } [get\_ports { out\_trg\_0 }]; #IO\_L22N\_T3\_AD7N\_35 Sch=led4\_b

#set\_property -dict { PACKAGE\_PIN G17 IOSTANDARD LVCMOS33 } [get\_ports { led\_0\_5 }]; #IO\_L16P\_T2\_35 Sch=led4\_g

#set\_property -dict { PACKAGE\_PIN N15 IOSTANDARD LVCMOS33 } [get\_ports { led\_0\_6 }]; #IO\_L21P\_T3\_DQS\_AD14P\_35 Sch=led4\_r

#set\_property -dict { PACKAGE\_PIN G14 IOSTANDARD LVCMOS33 } [get\_ports { led\_0[0] }]; #IO\_0\_35 Sch=led5\_b

#set\_property -dict { PACKAGE\_PIN L14 IOSTANDARD LVCMOS33 } [get\_ports { led\_0[1] }]; #IO\_L22P\_T3\_AD7P\_35 Sch=led5\_g

#set\_property -dict { PACKAGE\_PIN M15 IOSTANDARD LVCMOS33 } [get\_ports { led\_0[2] }]; #IO\_L23N\_T3\_35 Sch=led5\_r

##LEDs

#set\_property -dict { PACKAGE\_PIN R14 IOSTANDARD LVCMOS33 } [get\_ports { led\_0[0] }]; #IO\_L6N\_T0\_VREF\_34 Sch=led[0]

#set\_property -dict { PACKAGE\_PIN P14 IOSTANDARD LVCMOS33 } [get\_ports { led\_0[1] }]; #IO\_L6P\_T0\_34 Sch=led[1]

#set\_property -dict { PACKAGE\_PIN N16 IOSTANDARD LVCMOS33 } [get\_ports { led\_0[2] }]; #IO\_L21N\_T3\_DQS\_AD14N\_35 Sch=led[2]

#set\_property -dict { PACKAGE\_PIN M14 IOSTANDARD LVCMOS33 } [get\_ports { led\_0[3] }]; #IO\_L23P\_T3\_35 Sch=led[3]

##Buttons

#set\_property -dict { PACKAGE\_PIN D19 IOSTANDARD LVCMOS33 } [get\_ports { in\_sig\_0 }]; #IO\_L4P\_T0\_35 Sch=btn[0]

#set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets { in\_sig\_0\_IBUF }];

#set\_property -dict { PACKAGE\_PIN D20 IOSTANDARD LVCMOS33 } [get\_ports { fifo\_en\_0 }]; #IO\_L4N\_T0\_35 Sch=btn[1]

#set\_property -dict { PACKAGE\_PIN L20 IOSTANDARD LVCMOS33 } [get\_ports { btn[2] }]; #IO\_L9N\_T1\_DQS\_AD3N\_35 Sch=btn[2]

#set\_property -dict { PACKAGE\_PIN L19 IOSTANDARD LVCMOS33 } [get\_ports { btn[3] }]; #IO\_L9P\_T1\_DQS\_AD3P\_35 Sch=btn[3]

##Pmod Header JA

#set\_property -dict { PACKAGE\_PIN Y18 IOSTANDARD LVCMOS33 } [get\_ports { ja[0] }]; #IO\_L17P\_T2\_34 Sch=ja\_p[1]

#set\_property -dict { PACKAGE\_PIN Y19 IOSTANDARD LVCMOS33 } [get\_ports { ja[1] }]; #IO\_L17N\_T2\_34 Sch=ja\_n[1]

#set\_property -dict { PACKAGE\_PIN Y16 IOSTANDARD LVCMOS33 } [get\_ports { ja[2] }]; #IO\_L7P\_T1\_34 Sch=ja\_p[2]

#set\_property -dict { PACKAGE\_PIN Y17 IOSTANDARD LVCMOS33 } [get\_ports { ja[3] }]; #IO\_L7N\_T1\_34 Sch=ja\_n[2]

#set\_property -dict { PACKAGE\_PIN U18 IOSTANDARD LVCMOS33 } [get\_ports { ja[4] }]; #IO\_L12P\_T1\_MRCC\_34 Sch=ja\_p[3]

#set\_property -dict { PACKAGE\_PIN U19 IOSTANDARD LVCMOS33 } [get\_ports { ja[5] }]; #IO\_L12N\_T1\_MRCC\_34 Sch=ja\_n[3]

#set\_property -dict { PACKAGE\_PIN W18 IOSTANDARD LVCMOS33 } [get\_ports { ja[6] }]; #IO\_L22P\_T3\_34 Sch=ja\_p[4]

#set\_property -dict { PACKAGE\_PIN W19 IOSTANDARD LVCMOS33 } [get\_ports { ja[7] }]; #IO\_L22N\_T3\_34 Sch=ja\_n[4]

##Pmod Header JB

#set\_property -dict { PACKAGE\_PIN W14 IOSTANDARD LVCMOS33 } [get\_ports { jb[0] }]; #IO\_L8P\_T1\_34 Sch=jb\_p[1]

#set\_property -dict { PACKAGE\_PIN Y14 IOSTANDARD LVCMOS33 } [get\_ports { jb[1] }]; #IO\_L8N\_T1\_34 Sch=jb\_n[1]

#set\_property -dict { PACKAGE\_PIN T11 IOSTANDARD LVCMOS33 } [get\_ports { jb[2] }]; #IO\_L1P\_T0\_34 Sch=jb\_p[2]

#set\_property -dict { PACKAGE\_PIN T10 IOSTANDARD LVCMOS33 } [get\_ports { jb[3] }]; #IO\_L1N\_T0\_34 Sch=jb\_n[2]

#set\_property -dict { PACKAGE\_PIN V16 IOSTANDARD LVCMOS33 } [get\_ports { jb[4] }]; #IO\_L18P\_T2\_34 Sch=jb\_p[3]

#set\_property -dict { PACKAGE\_PIN W16 IOSTANDARD LVCMOS33 } [get\_ports { jb[5] }]; #IO\_L18N\_T2\_34 Sch=jb\_n[3]

#set\_property -dict { PACKAGE\_PIN V12 IOSTANDARD LVCMOS33 } [get\_ports { jb[6] }]; #IO\_L4P\_T0\_34 Sch=jb\_p[4]

#set\_property -dict { PACKAGE\_PIN W13 IOSTANDARD LVCMOS33 } [get\_ports { jb[7] }]; #IO\_L4N\_T0\_34 Sch=jb\_n[4]

##Audio Out

#set\_property -dict { PACKAGE\_PIN R18 IOSTANDARD LVCMOS33 } [get\_ports { aud\_pwm }]; #IO\_L20N\_T3\_34 Sch=aud\_pwm

#set\_property -dict { PACKAGE\_PIN T17 IOSTANDARD LVCMOS33 } [get\_ports { aud\_sd }]; #IO\_L20P\_T3\_34 Sch=aud\_sd

##Mic input

#set\_property -dict { PACKAGE\_PIN F17 IOSTANDARD LVCMOS33 } [get\_ports { m\_clk }]; #IO\_L6N\_T0\_VREF\_35 Sch=m\_clk

#set\_property -dict { PACKAGE\_PIN G18 IOSTANDARD LVCMOS33 } [get\_ports { m\_data }]; #IO\_L16N\_T2\_35 Sch=m\_data

##ChipKit Single Ended Analog Inputs

##NOTE: The ck\_an\_p pins can be used as single ended analog inputs with voltages from 0-3.3V (Chipkit Analog pins A0-A5).

## These signals should only be connected to the XADC core. When using these pins as digital I/O, use pins ck\_io[14-19].

set\_property -dict { PACKAGE\_PIN D18 IOSTANDARD LVCMOS33 } [get\_ports { Vaux1\_0\_v\_n }]; #IO\_L3N\_T0\_DQS\_AD1N\_35 Sch=ck\_an\_n[0]

set\_property -dict { PACKAGE\_PIN E17 IOSTANDARD LVCMOS33 } [get\_ports { Vaux1\_0\_v\_p }]; #IO\_L3P\_T0\_DQS\_AD1P\_35 Sch=ck\_an\_p[0]

#set\_property -dict { PACKAGE\_PIN E19 IOSTANDARD LVCMOS33 } [get\_ports { Vaux9\_0\_v\_n }]; #IO\_L5N\_T0\_AD9N\_35 Sch=ck\_an\_n[1]

#set\_property -dict { PACKAGE\_PIN E18 IOSTANDARD LVCMOS33 } [get\_ports { Vaux9\_0\_v\_p }]; #IO\_L5P\_T0\_AD9P\_35 Sch=ck\_an\_p[1]

#set\_property -dict { PACKAGE\_PIN J14 IOSTANDARD LVCMOS33 } [get\_ports { Vaux6\_0\_v\_n }]; #IO\_L20N\_T3\_AD6N\_35 Sch=ck\_an\_n[2]

#set\_property -dict { PACKAGE\_PIN K14 IOSTANDARD LVCMOS33 } [get\_ports { Vaux6\_0\_v\_p }]; #IO\_L20P\_T3\_AD6P\_35 Sch=ck\_an\_p[2]

#set\_property -dict { PACKAGE\_PIN J16 IOSTANDARD LVCMOS33 } [get\_ports { Vaux15\_0\_v\_n }]; #IO\_L24N\_T3\_AD15N\_35 Sch=ck\_an\_n[3]

#set\_property -dict { PACKAGE\_PIN K16 IOSTANDARD LVCMOS33 } [get\_ports { Vaux15\_0\_v\_p }]; #IO\_L24P\_T3\_AD15P\_35 Sch=ck\_an\_p[3]

#set\_property -dict { PACKAGE\_PIN H20 IOSTANDARD LVCMOS33 } [get\_ports { ck\_an\_n[4] }]; #IO\_L17N\_T2\_AD5N\_35 Sch=ck\_an\_n[4]

#set\_property -dict { PACKAGE\_PIN J20 IOSTANDARD LVCMOS33 } [get\_ports { ck\_an\_p[4] }]; #IO\_L17P\_T2\_AD5P\_35 Sch=ck\_an\_p[4]

#set\_property -dict { PACKAGE\_PIN G20 IOSTANDARD LVCMOS33 } [get\_ports { ck\_an\_n[5] }]; #IO\_L18N\_T2\_AD13N\_35 Sch=ck\_an\_n[5]

#set\_property -dict { PACKAGE\_PIN G19 IOSTANDARD LVCMOS33 } [get\_ports { ck\_an\_p[5] }]; #IO\_L18P\_T2\_AD13P\_35 Sch=ck\_an\_p[5]

##ChipKit Digital I/O Low

#set\_property -dict { PACKAGE\_PIN T14 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[0] }]; #IO\_L5P\_T0\_34 Sch=ck\_io[0]

#set\_property -dict { PACKAGE\_PIN U12 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[1] }]; #IO\_L2N\_T0\_34 Sch=ck\_io[1]

#set\_property -dict { PACKAGE\_PIN U13 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[2] }]; #IO\_L3P\_T0\_DQS\_PUDC\_B\_34 Sch=ck\_io[2]

#set\_property -dict { PACKAGE\_PIN V13 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[3] }]; #IO\_L3N\_T0\_DQS\_34 Sch=ck\_io[3]

#set\_property -dict { PACKAGE\_PIN V15 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[4] }]; #IO\_L10P\_T1\_34 Sch=ck\_io[4]

#set\_property -dict { PACKAGE\_PIN T15 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[5] }]; #IO\_L5N\_T0\_34 Sch=ck\_io[5]

#set\_property -dict { PACKAGE\_PIN R16 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[6] }]; #IO\_L19P\_T3\_34 Sch=ck\_io[6]

#set\_property -dict { PACKAGE\_PIN U17 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[7] }]; #IO\_L9N\_T1\_DQS\_34 Sch=ck\_io[7]

#set\_property -dict { PACKAGE\_PIN V17 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[8] }]; #IO\_L21P\_T3\_DQS\_34 Sch=ck\_io[8]

#set\_property -dict { PACKAGE\_PIN V18 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[9] }]; #IO\_L21N\_T3\_DQS\_34 Sch=ck\_io[9]

#set\_property -dict { PACKAGE\_PIN T16 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[10] }]; #IO\_L9P\_T1\_DQS\_34 Sch=ck\_io[10]

#set\_property -dict { PACKAGE\_PIN R17 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[11] }]; #IO\_L19N\_T3\_VREF\_34 Sch=ck\_io[11]

#set\_property -dict { PACKAGE\_PIN P18 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[12] }]; #IO\_L23N\_T3\_34 Sch=ck\_io[12]

#set\_property -dict { PACKAGE\_PIN N17 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[13] }]; #IO\_L23P\_T3\_34 Sch=ck\_io[13]

##ChipKit Digital I/O On Outer Analog Header

##NOTE: These pins should be used when using the analog header signals A0-A5 as digital I/O (Chipkit digital pins 14-19)

#set\_property -dict { PACKAGE\_PIN Y11 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[14] }]; #IO\_L18N\_T2\_13 Sch=ck\_a[0]

#set\_property -dict { PACKAGE\_PIN Y12 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[15] }]; #IO\_L20P\_T3\_13 Sch=ck\_a[1]

#set\_property -dict { PACKAGE\_PIN W11 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[16] }]; #IO\_L18P\_T2\_13 Sch=ck\_a[2]

#set\_property -dict { PACKAGE\_PIN V11 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[17] }]; #IO\_L21P\_T3\_DQS\_13 Sch=ck\_a[3]

#set\_property -dict { PACKAGE\_PIN T5 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[18] }]; #IO\_L19P\_T3\_13 Sch=ck\_a[4]

#set\_property -dict { PACKAGE\_PIN U10 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[19] }]; #IO\_L12N\_T1\_MRCC\_13 Sch=ck\_a[5]

##ChipKit Digital I/O On Inner Analog Header

##NOTE: These pins will need to be connected to the XADC core when used as differential analog inputs (Chipkit analog pins A6-A11)

#set\_property -dict { PACKAGE\_PIN B20 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[20] }]; #IO\_L1N\_T0\_AD0N\_35 Sch=ad\_n[0]

#set\_property -dict { PACKAGE\_PIN C20 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[21] }]; #IO\_L1P\_T0\_AD0P\_35 Sch=ad\_p[0]

#set\_property -dict { PACKAGE\_PIN F20 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[22] }]; #IO\_L15N\_T2\_DQS\_AD12N\_35 Sch=ad\_n[12]

#set\_property -dict { PACKAGE\_PIN F19 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[23] }]; #IO\_L15P\_T2\_DQS\_AD12P\_35 Sch=ad\_p[12]

#set\_property -dict { PACKAGE\_PIN A20 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[24] }]; #IO\_L2N\_T0\_AD8N\_35 Sch=ad\_n[8]

#set\_property -dict { PACKAGE\_PIN B19 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[25] }]; #IO\_L2P\_T0\_AD8P\_35 Sch=ad\_p[8]

##ChipKit Digital I/O High

#set\_property -dict { PACKAGE\_PIN U5 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[26] }]; #IO\_L19N\_T3\_VREF\_13 Sch=ck\_io[26]

#set\_property -dict { PACKAGE\_PIN V5 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[27] }]; #IO\_L6N\_T0\_VREF\_13 Sch=ck\_io[27]

#set\_property -dict { PACKAGE\_PIN V6 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[28] }]; #IO\_L22P\_T3\_13 Sch=ck\_io[28]

#set\_property -dict { PACKAGE\_PIN U7 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[29] }]; #IO\_L11P\_T1\_SRCC\_13 Sch=ck\_io[29]

#set\_property -dict { PACKAGE\_PIN V7 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[30] }]; #IO\_L11N\_T1\_SRCC\_13 Sch=ck\_io[30]

#set\_property -dict { PACKAGE\_PIN U8 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[31] }]; #IO\_L17N\_T2\_13 Sch=ck\_io[31]

#set\_property -dict { PACKAGE\_PIN V8 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[32] }]; #IO\_L15P\_T2\_DQS\_13 Sch=ck\_io[32]

#set\_property -dict { PACKAGE\_PIN V10 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[33] }]; #IO\_L21N\_T3\_DQS\_13 Sch=ck\_io[33]

#set\_property -dict { PACKAGE\_PIN W10 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[34] }]; #IO\_L16P\_T2\_13 Sch=ck\_io[34]

#set\_property -dict { PACKAGE\_PIN W6 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[35] }]; #IO\_L22N\_T3\_13 Sch=ck\_io[35]

#set\_property -dict { PACKAGE\_PIN Y6 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[36] }]; #IO\_L13N\_T2\_MRCC\_13 Sch=ck\_io[36]

#set\_property -dict { PACKAGE\_PIN Y7 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[37] }]; #IO\_L13P\_T2\_MRCC\_13 Sch=ck\_io[37]

#set\_property -dict { PACKAGE\_PIN W8 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[38] }]; #IO\_L15N\_T2\_DQS\_13 Sch=ck\_io[38]

#set\_property -dict { PACKAGE\_PIN Y8 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[39] }]; #IO\_L14N\_T2\_SRCC\_13 Sch=ck\_io[39]

#set\_property -dict { PACKAGE\_PIN W9 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[40] }]; #IO\_L16N\_T2\_13 Sch=ck\_io[40]

#set\_property -dict { PACKAGE\_PIN Y9 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[41] }]; #IO\_L14P\_T2\_SRCC\_13 Sch=ck\_io[41]

#set\_property -dict { PACKAGE\_PIN Y13 IOSTANDARD LVCMOS33 } [get\_ports { ck\_io[42] }]; #IO\_L20N\_T3\_13 Sch=ck\_ioa

## ChipKit SPI

#set\_property -dict { PACKAGE\_PIN W15 IOSTANDARD LVCMOS33 } [get\_ports { ck\_miso }]; #IO\_L10N\_T1\_34 Sch=ck\_miso

#set\_property -dict { PACKAGE\_PIN T12 IOSTANDARD LVCMOS33 } [get\_ports { ck\_mosi }]; #IO\_L2P\_T0\_34 Sch=ck\_mosi

#set\_property -dict { PACKAGE\_PIN H15 IOSTANDARD LVCMOS33 } [get\_ports { ck\_sck }]; #IO\_L19P\_T3\_35 Sch=ck\_sck

#set\_property -dict { PACKAGE\_PIN F16 IOSTANDARD LVCMOS33 } [get\_ports { ck\_ss }]; #IO\_L6P\_T0\_35 Sch=ck\_ss

## ChipKit I2C

#set\_property -dict { PACKAGE\_PIN P16 IOSTANDARD LVCMOS33 } [get\_ports { ck\_scl }]; #IO\_L24N\_T3\_34 Sch=ck\_scl

#set\_property -dict { PACKAGE\_PIN P15 IOSTANDARD LVCMOS33 } [get\_ports { ck\_sda }]; #IO\_L24P\_T3\_34 Sch=ck\_sda

##HDMI Rx

#set\_property -dict { PACKAGE\_PIN H17 IOSTANDARD LVCMOS33 } [get\_ports { hdmi\_rx\_cec }]; #IO\_L13N\_T2\_MRCC\_35 Sch=hdmi\_rx\_cec

#set\_property -dict { PACKAGE\_PIN P19 IOSTANDARD TMDS\_33 } [get\_ports { hdmi\_rx\_clk\_n }]; #IO\_L13N\_T2\_MRCC\_34 Sch=hdmi\_rx\_clk\_n

#set\_property -dict { PACKAGE\_PIN N18 IOSTANDARD TMDS\_33 } [get\_ports { hdmi\_rx\_clk\_p }]; #IO\_L13P\_T2\_MRCC\_34 Sch=hdmi\_rx\_clk\_p

#set\_property -dict { PACKAGE\_PIN W20 IOSTANDARD TMDS\_33 } [get\_ports { hdmi\_rx\_d\_n[0] }]; #IO\_L16N\_T2\_34 Sch=hdmi\_rx\_d\_n[0]

#set\_property -dict { PACKAGE\_PIN V20 IOSTANDARD TMDS\_33 } [get\_ports { hdmi\_rx\_d\_p[0] }]; #IO\_L16P\_T2\_34 Sch=hdmi\_rx\_d\_p[0]

#set\_property -dict { PACKAGE\_PIN U20 IOSTANDARD TMDS\_33 } [get\_ports { hdmi\_rx\_d\_n[1] }]; #IO\_L15N\_T2\_DQS\_34 Sch=hdmi\_rx\_d\_n[1]

#set\_property -dict { PACKAGE\_PIN T20 IOSTANDARD TMDS\_33 } [get\_ports { hdmi\_rx\_d\_p[1] }]; #IO\_L15P\_T2\_DQS\_34 Sch=hdmi\_rx\_d\_p[1]

#set\_property -dict { PACKAGE\_PIN P20 IOSTANDARD TMDS\_33 } [get\_ports { hdmi\_rx\_d\_n[2] }]; #IO\_L14N\_T2\_SRCC\_34 Sch=hdmi\_rx\_d\_n[2]

#set\_property -dict { PACKAGE\_PIN N20 IOSTANDARD TMDS\_33 } [get\_ports { hdmi\_rx\_d\_p[2] }]; #IO\_L14P\_T2\_SRCC\_34 Sch=hdmi\_rx\_d\_p[2]

#set\_property -dict { PACKAGE\_PIN T19 IOSTANDARD LVCMOS33 } [get\_ports { hdmi\_rx\_hpd }]; #IO\_25\_34 Sch=hdmi\_rx\_hpd

#set\_property -dict { PACKAGE\_PIN U14 IOSTANDARD LVCMOS33 } [get\_ports { hdmi\_rx\_scl }]; #IO\_L11P\_T1\_SRCC\_34 Sch=hdmi\_rx\_scl

#set\_property -dict { PACKAGE\_PIN U15 IOSTANDARD LVCMOS33 } [get\_ports { hdmi\_rx\_sda }]; #IO\_L11N\_T1\_SRCC\_34 Sch=hdmi\_rx\_sda

##HDMI Tx

#set\_property -dict { PACKAGE\_PIN G15 IOSTANDARD LVCMOS33 } [get\_ports { hdmi\_tx\_cec }]; #IO\_L19N\_T3\_VREF\_35 Sch=hdmi\_tx\_cec

#set\_property -dict { PACKAGE\_PIN L17 IOSTANDARD TMDS\_33 } [get\_ports { hdmi\_tx\_clk\_n }]; #IO\_L11N\_T1\_SRCC\_35 Sch=hdmi\_tx\_clk\_n

#set\_property -dict { PACKAGE\_PIN L16 IOSTANDARD TMDS\_33 } [get\_ports { hdmi\_tx\_clk\_p }]; #IO\_L11P\_T1\_SRCC\_35 Sch=hdmi\_tx\_clk\_p

#set\_property -dict { PACKAGE\_PIN K18 IOSTANDARD TMDS\_33 } [get\_ports { hdmi\_tx\_d\_n[0] }]; #IO\_L12N\_T1\_MRCC\_35 Sch=hdmi\_tx\_d\_n[0]

#set\_property -dict { PACKAGE\_PIN K17 IOSTANDARD TMDS\_33 } [get\_ports { hdmi\_tx\_d\_p[0] }]; #IO\_L12P\_T1\_MRCC\_35 Sch=hdmi\_tx\_d\_p[0]

#set\_property -dict { PACKAGE\_PIN J19 IOSTANDARD TMDS\_33 } [get\_ports { hdmi\_tx\_d\_n[1] }]; #IO\_L10N\_T1\_AD11N\_35 Sch=hdmi\_tx\_d\_n[1]

#set\_property -dict { PACKAGE\_PIN K19 IOSTANDARD TMDS\_33 } [get\_ports { hdmi\_tx\_d\_p[1] }]; #IO\_L10P\_T1\_AD11P\_35 Sch=hdmi\_tx\_d\_p[1]

#set\_property -dict { PACKAGE\_PIN H18 IOSTANDARD TMDS\_33 } [get\_ports { hdmi\_tx\_d\_n[2] }]; #IO\_L14N\_T2\_AD4N\_SRCC\_35 Sch=hdmi\_tx\_d\_n[2]

#set\_property -dict { PACKAGE\_PIN J18 IOSTANDARD TMDS\_33 } [get\_ports { hdmi\_tx\_d\_p[2] }]; #IO\_L14P\_T2\_AD4P\_SRCC\_35 Sch=hdmi\_tx\_d\_p[2]

#set\_property -dict { PACKAGE\_PIN R19 IOSTANDARD LVCMOS33 } [get\_ports { hdmi\_tx\_hpdn }]; #IO\_0\_34 Sch=hdmi\_tx\_hpdn

#set\_property -dict { PACKAGE\_PIN M17 IOSTANDARD LVCMOS33 } [get\_ports { hdmi\_tx\_scl }]; #IO\_L8P\_T1\_AD10P\_35 Sch=hdmi\_tx\_scl

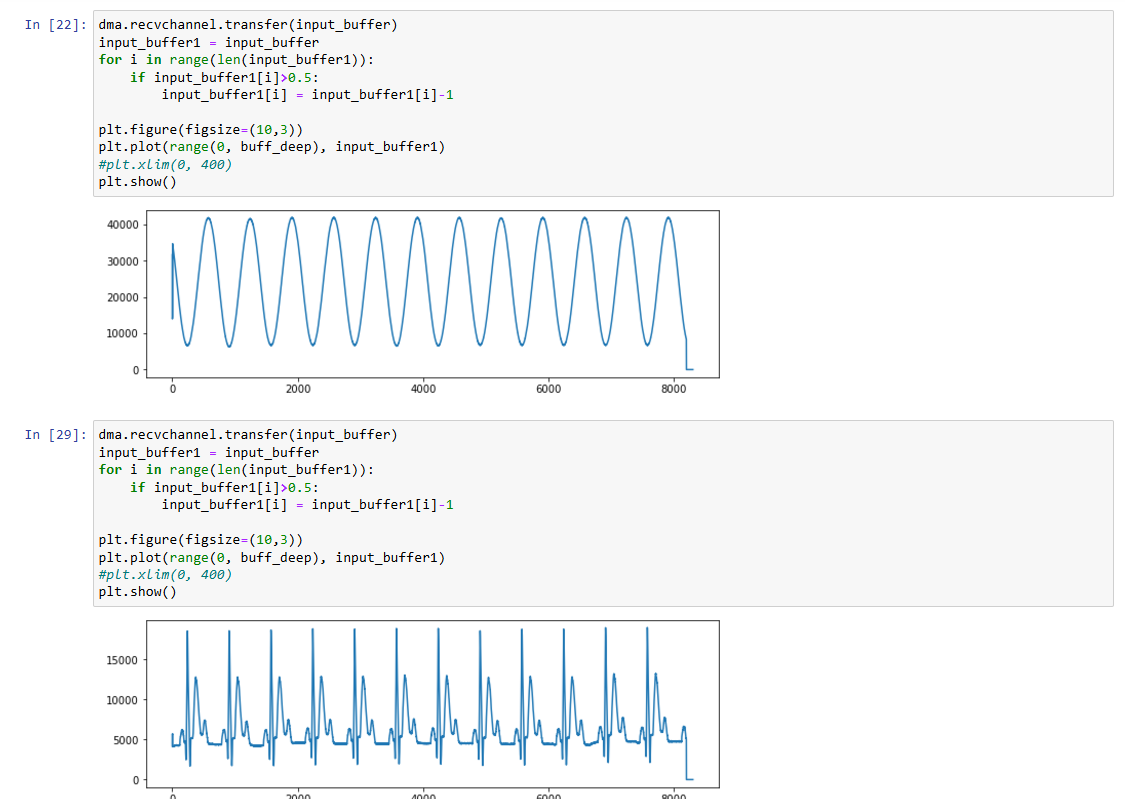
#set\_property -dict { PACKAGE\_PIN M18 IOSTANDARD LVCMOS33 } [get\_ports { hdmi\_tx\_sda }]; #IO\_L8N\_T1\_AD10N\_35 Sch=hdmi\_tx\_sda

##Crypto SDA

#set\_property -dict { PACKAGE\_PIN J15 IOSTANDARD LVCMOS33 } [get\_ports { crypto\_sda }]; #IO\_25\_35 Sch=crypto\_sda

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



from pynq import PL

from pynq import Overlay

from pynq import allocate

import numpy as np

import matplotlib.pyplot as plt

PL.reset()

xadc\_stream = Overlay('xstream.bit')

dma = xadc\_stream.axi\_dma\_0

buff\_deep = 8300

input\_buffer = allocate(shape=(buff\_deep,), dtype=np.uint32)

dma.recvchannel.transfer(input\_buffer)

input\_buffer1 = input\_buffer

for i in range(len(input\_buffer1)):

if input\_buffer1[i]>0.5:

input\_buffer1[i] = input\_buffer1[i]-1

plt.figure(figsize=(10,3))

plt.plot(range(0, buff\_deep), input\_buffer1)

#plt.xlim(0, 400)

plt.show()