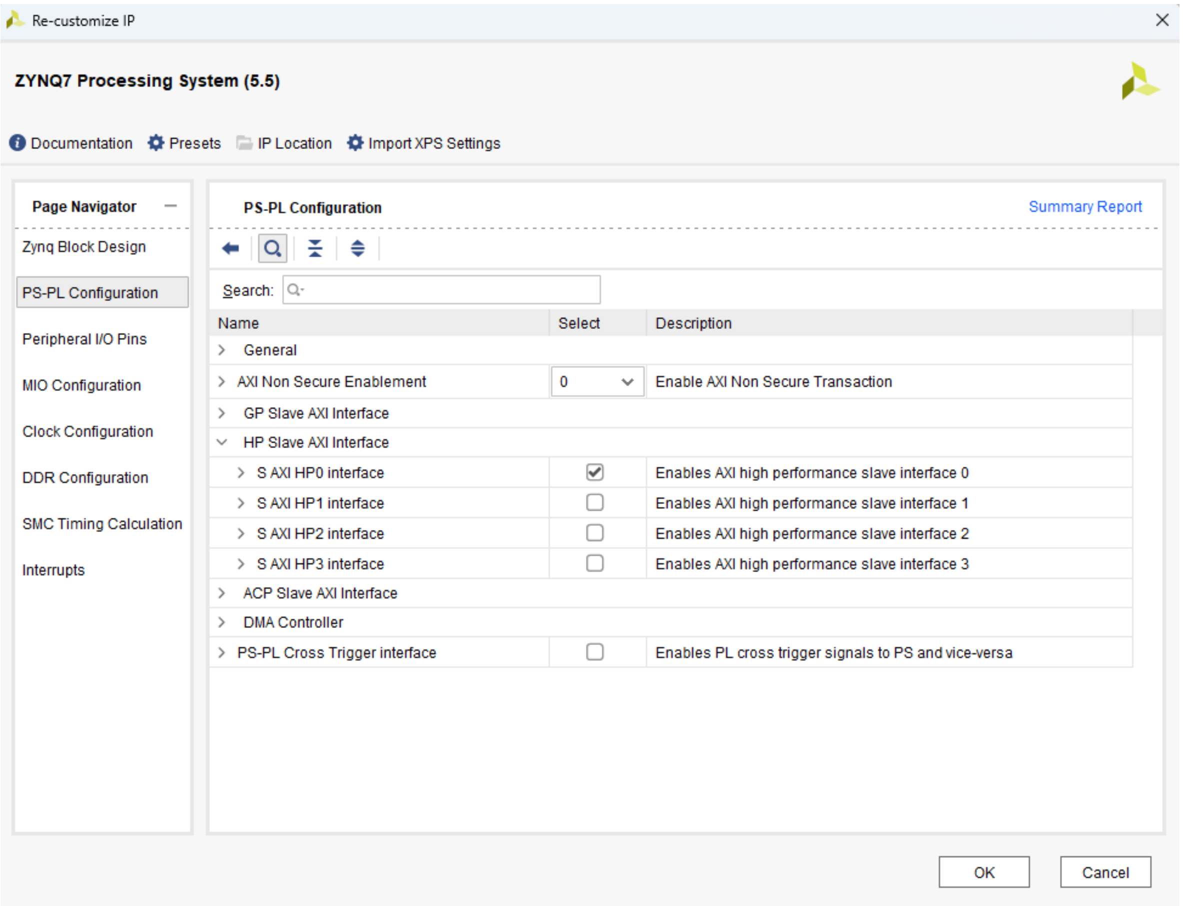
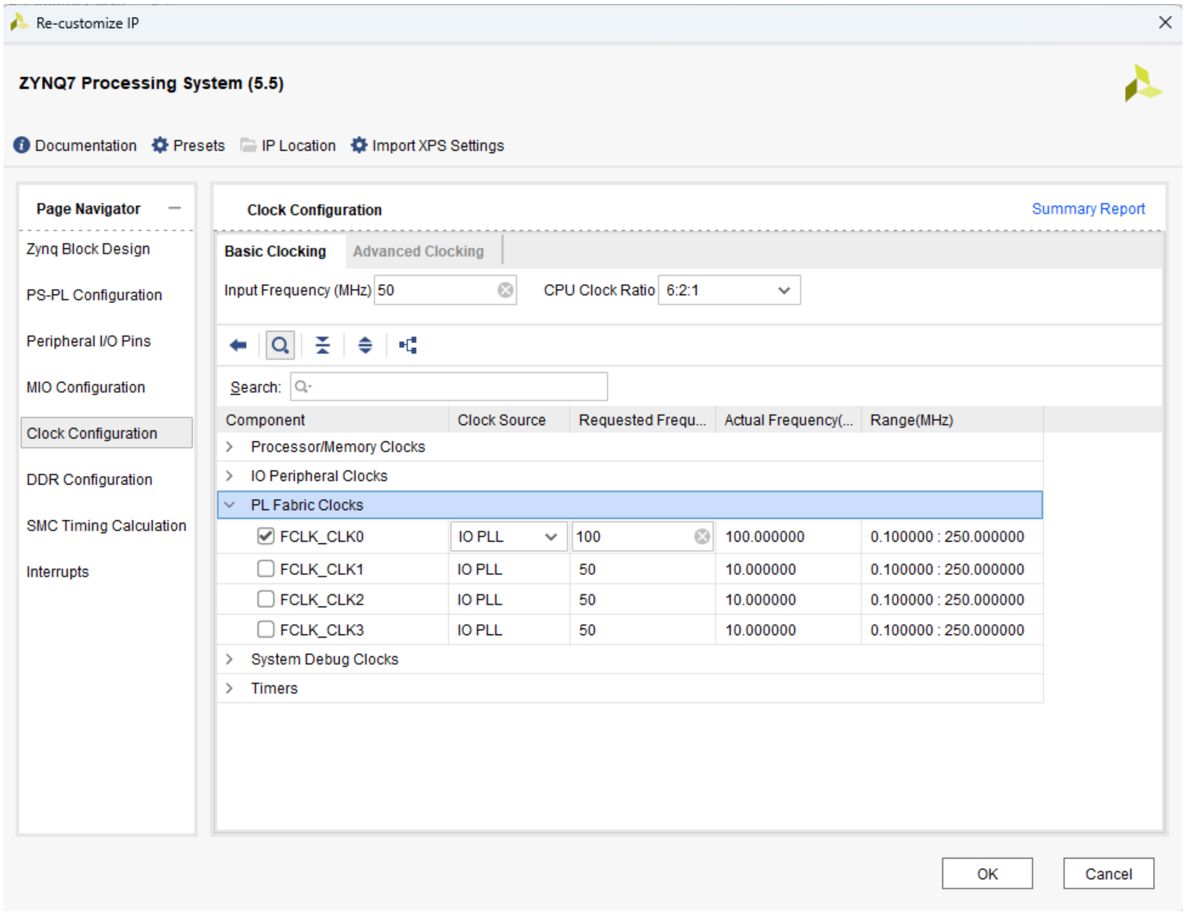
Creacion de AXI Stream

Implementación del ZYNQ:



PS-PL Configuration

Name	Select	Description
› General		
› AXI Non Secure Enablement	0	Enable AXI Non Secure Transaction
› GP Slave AXI Interface		
⌄ HP Slave AXI Interface		
› S AXI HP0 interface	☑	Enables AXI high performance slave interface 0
› S AXI HP1 interface	☐	Enables AXI high performance slave interface 1
› S AXI HP2 interface	☐	Enables AXI high performance slave interface 2
› S AXI HP3 interface	☐	Enables AXI high performance slave interface 3
› ACP Slave AXI Interface		
› DMA Controller		
› PS-PL Cross Trigger interface	☐	Enables PL cross trigger signals to PS and vice-versa



ZYNQ7 Processing System (5.5)

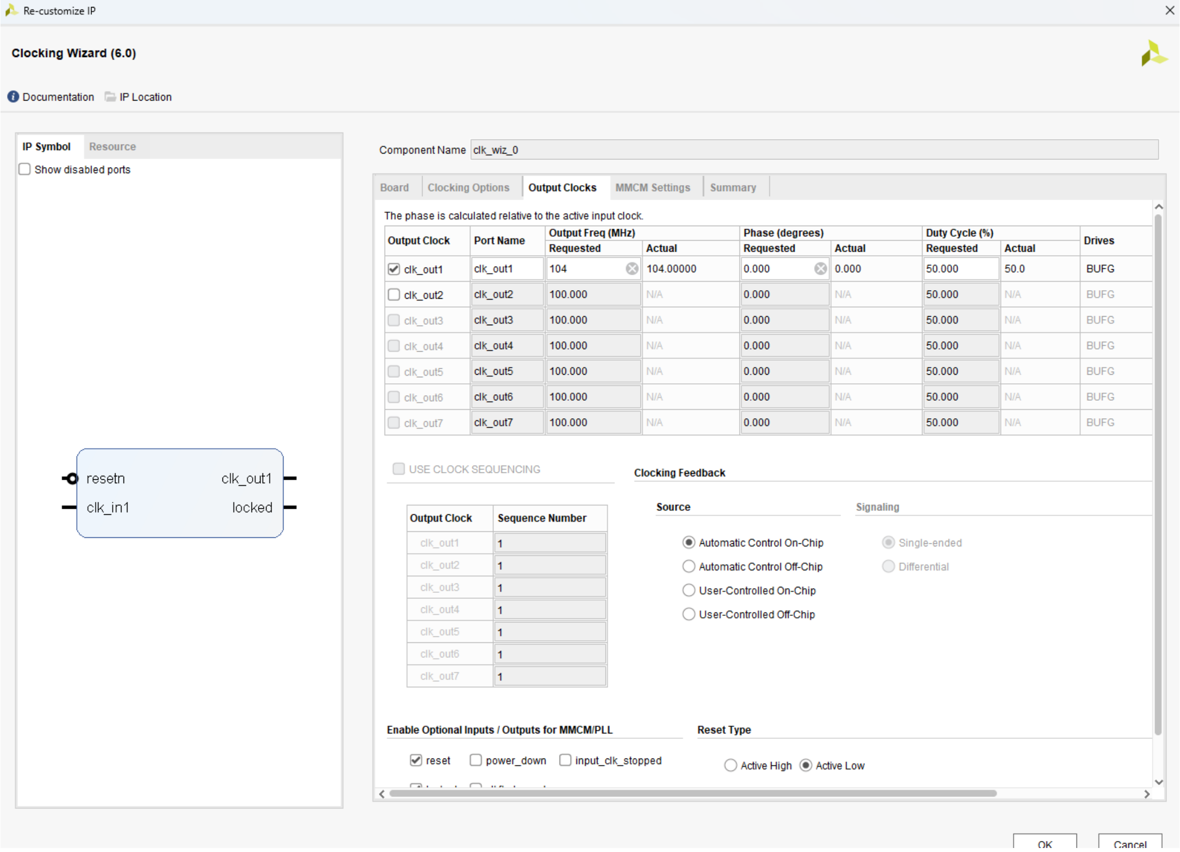
Clock Configuration

Input Frequency (MHz) 50

CPU Clock Ratio 6:2:1

Component	Clock Source	Requested Frequ...	Actual Frequency(...	Range(MHz)
› Processor/Memory Clocks				
› IO Peripheral Clocks				
⌄ PL Fabric Clocks				
☑ FCLK_CLK0	IO PLL	100	100.000000	0.100000 : 250.000000
□ FCLK_CLK1	IO PLL	50	10.000000	0.100000 : 250.000000
□ FCLK_CLK2	IO PLL	50	10.000000	0.100000 : 250.000000
□ FCLK_CLK3	IO PLL	50	10.000000	0.100000 : 250.000000
› System Debug Clocks				
› Timers				

Clock Wizard



☐ Show disabled ports

Component Name clk_wiz_0

The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)		Drives
		Requested	Actual	Requested	Actual	Requested	Actual	
☑ clk_out1	clk_out1	104	104.00000	0.000	0.000	50.000	50.0	BUFG
☐ clk_out2	clk_out2	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
☐ clk_out3	clk_out3	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
☐ clk_out4	clk_out4	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
☐ clk_out5	clk_out5	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
☐ clk_out6	clk_out6	100.000	N/A	0.000	N/A	50.000	N/A	BUFG
☐ clk_out7	clk_out7	100.000	N/A	0.000	N/A	50.000	N/A	BUFG

☐ USE CLOCK SEQUENCING

Output Clock	Sequence Number
clk_out1	1
clk_out2	1
clk_out3	1
clk_out4	1
clk_out5	1
clk_out6	1
clk_out7	1

Clocking Feedback

Source

● Automatic Control On-Chip
○ Automatic Control Off-Chip
○ User-Controlled On-Chip
○ User-Controlled Off-Chip

Signaling

● Single-ended
○ Differential

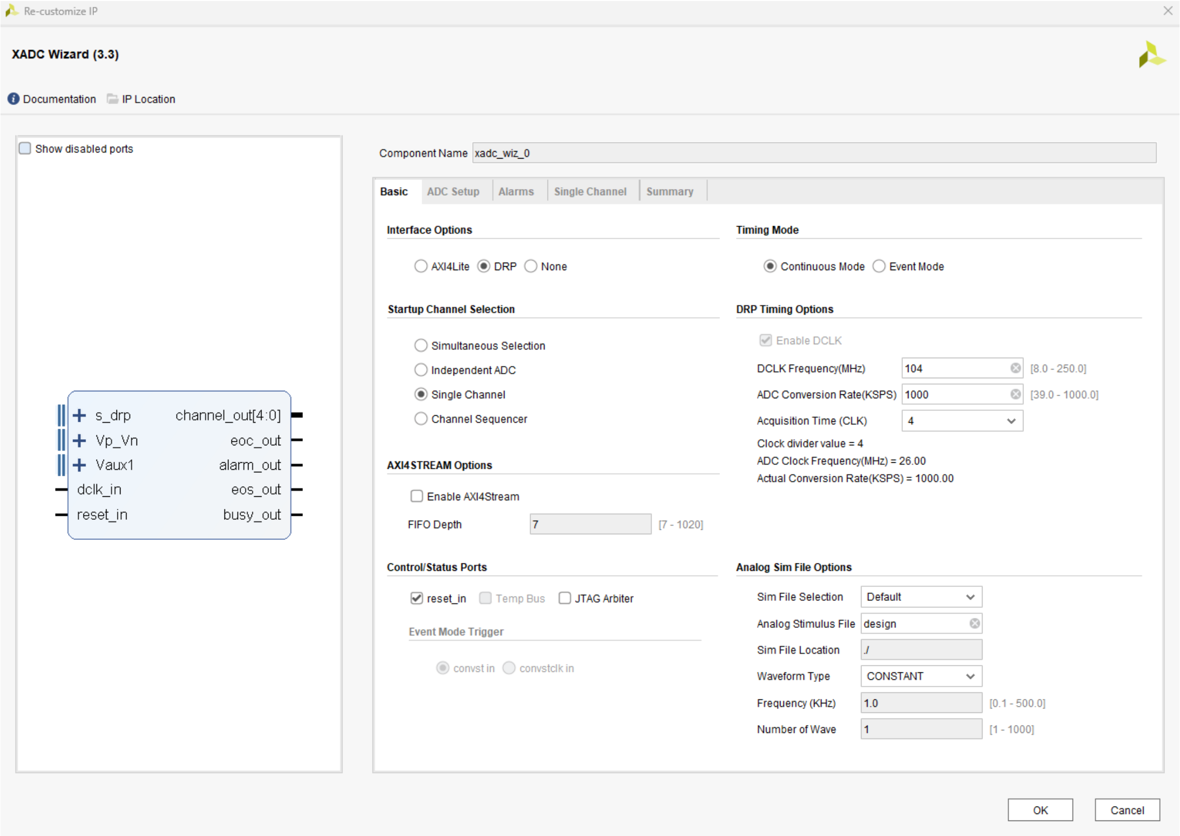
Enable Optional Inputs / Outputs for MMCM/PLL

☑ reset ☐ power_down ☐ input_clk_stopped

Reset Type

○ Active High ● Active Low

XADC



Re-customize IP

XADC Wizard (3.3)

Documentation   IP Location

□ Show disabled ports

s_drp   channel_out[4:0]
Vp_Vn   eoc_out
Vaux1   alarm_out
dclk_in   eos_out
reset_in   busy_out

Component Name: xadc_wiz_0

Basic | ADC Setup | Alarms | Single Channel | Summary

Interface Options

○ AXI4Lite ● DRP ○ None

Timing Mode

● Continuous Mode ○ Event Mode

Startup Channel Selection

○ Simultaneous Selection
○ Independent ADC
● Single Channel
○ Channel Sequencer

DRP Timing Options

☑ Enable DCLK

DCLK Frequency(MHz)	104	[8.0 - 250.0]
ADC Conversion Rate(KSPS)	1000	[39.0 - 1000.0]
Acquisition Time (CLK)	4	

Clock divider value = 4
ADC Clock Frequency(MHz) = 26.00
Actual Conversion Rate(KSPS) = 1000.00

AXI4STREAM Options

□ Enable AXI4Stream

FIFO Depth: 7 [7 - 1020]

Control/Status Ports

☑ reset_in □ Temp Bus □ JTAG Arbiter

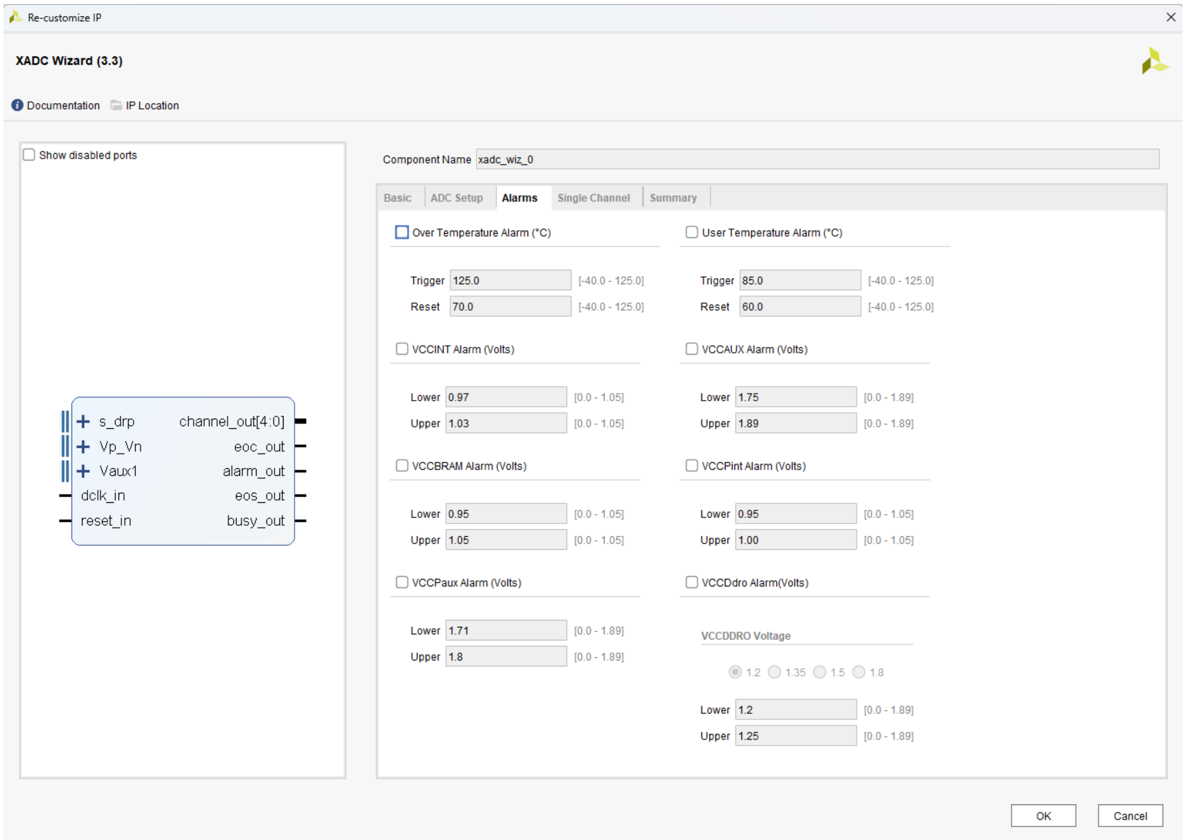
Event Mode Trigger

● convst in ○ convstclk in

Analog Sim File Options

Sim File Selection	Default	
Analog Stimulus File	design	
Sim File Location	./	
Waveform Type	CONSTANT	
Frequency (KHz)	1.0	[0.1 - 500.0]
Number of Wave	1	[1 - 1000]

OK   Cancel



Re-customize IP

XADC Wizard (3.3)

Documentation   IP Location

□ Show disabled ports

s_drp   channel_out[4:0]
Vp_Vn   eoc_out
Vaux1   alarm_out
dclk_in   eos_out
reset_in   busy_out

Component Name: xadc_wiz_0

Basic | ADC Setup | Alarms | Single Channel | Summary

□ Over Temperature Alarm (°C)

Trigger	125.0	[-40.0 - 125.0]
Reset	70.0	[-40.0 - 125.0]

□ User Temperature Alarm (°C)

Trigger	85.0	[-40.0 - 125.0]
Reset	60.0	[-40.0 - 125.0]

□ VCCINT Alarm (Volts)

Lower	0.97	[0.0 - 1.05]
Upper	1.03	[0.0 - 1.05]

□ VCCAUX Alarm (Volts)

Lower	1.75	[0.0 - 1.89]
Upper	1.89	[0.0 - 1.89]

□ VCCBRAM Alarm (Volts)

Lower	0.95	[0.0 - 1.05]
Upper	1.05	[0.0 - 1.05]

□ VCCPint Alarm (Volts)

Lower	0.95	[0.0 - 1.05]
Upper	1.00	[0.0 - 1.05]

□ VCCPaux Alarm (Volts)

Lower	1.71	[0.0 - 1.89]
Upper	1.8	[0.0 - 1.89]

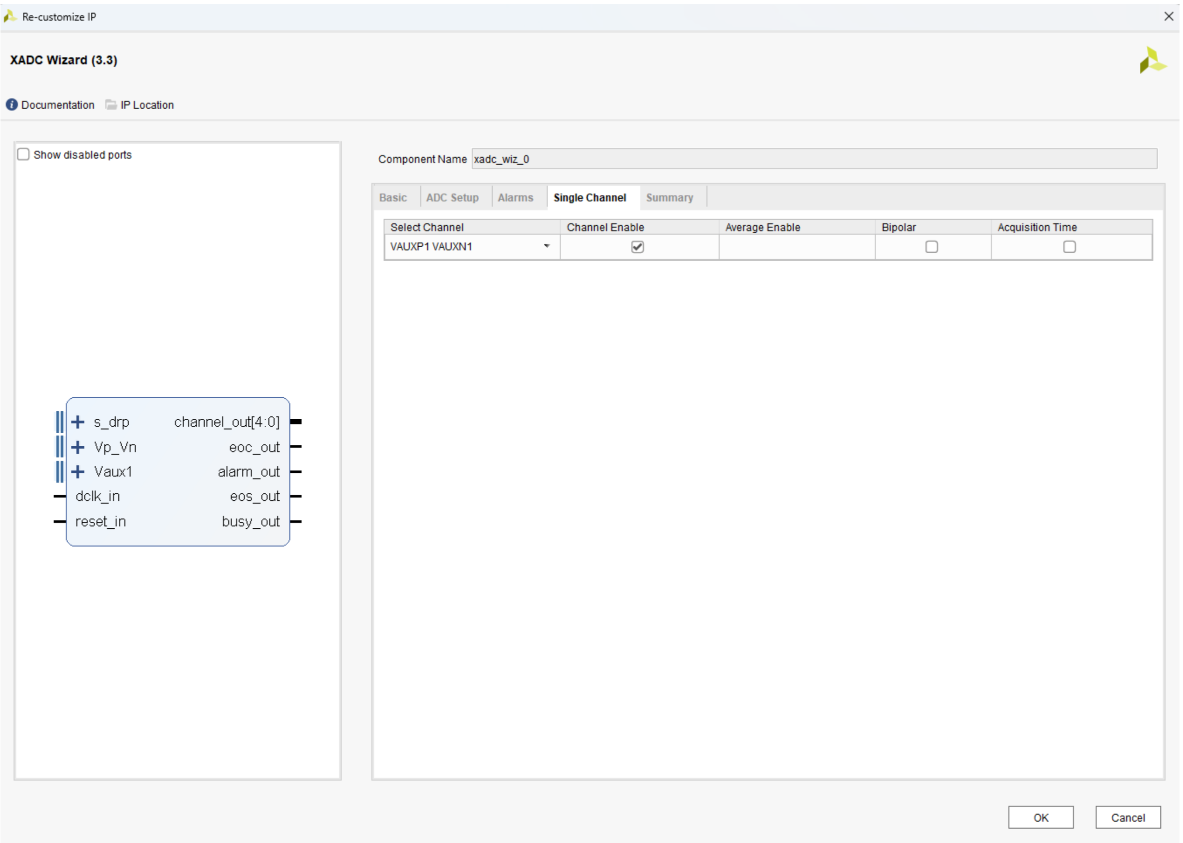
□ VCCDdro Alarm(Volts)

VCCDDRO Voltage

● 1.2 ○ 1.35 ○ 1.5 ○ 1.8

Lower	1.2	[0.0 - 1.89]
Upper	1.25	[0.0 - 1.89]

OK   Cancel



Re-customize IP

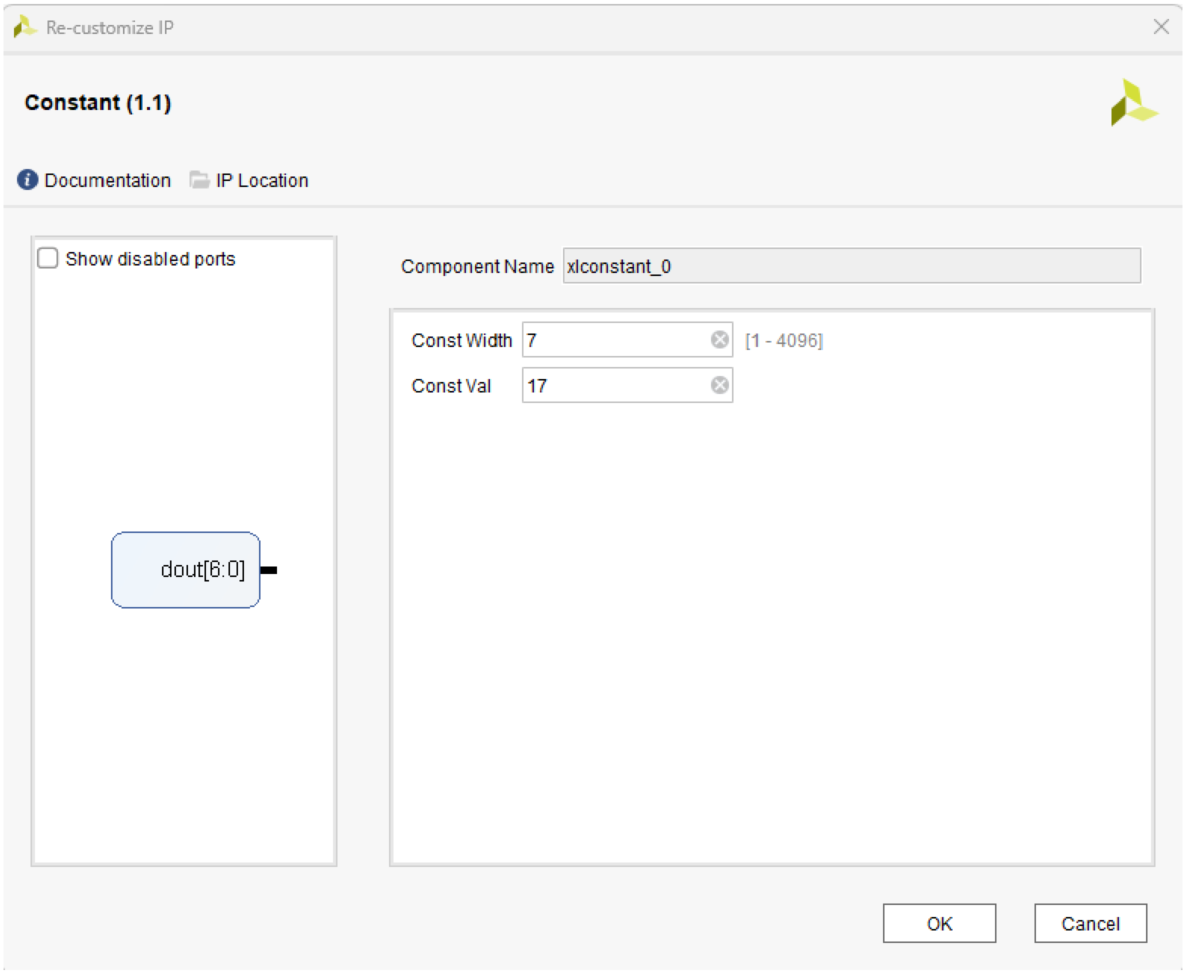
XADC Wizard (3.3)

☐ Show disabled ports

Component Name xadc_wiz_0

Select Channel	Channel Enable	Average Enable	Bipolar	Acquisition Time
VAUXP1 VAUXN1	☑		☐	☐

Xconstant para el address



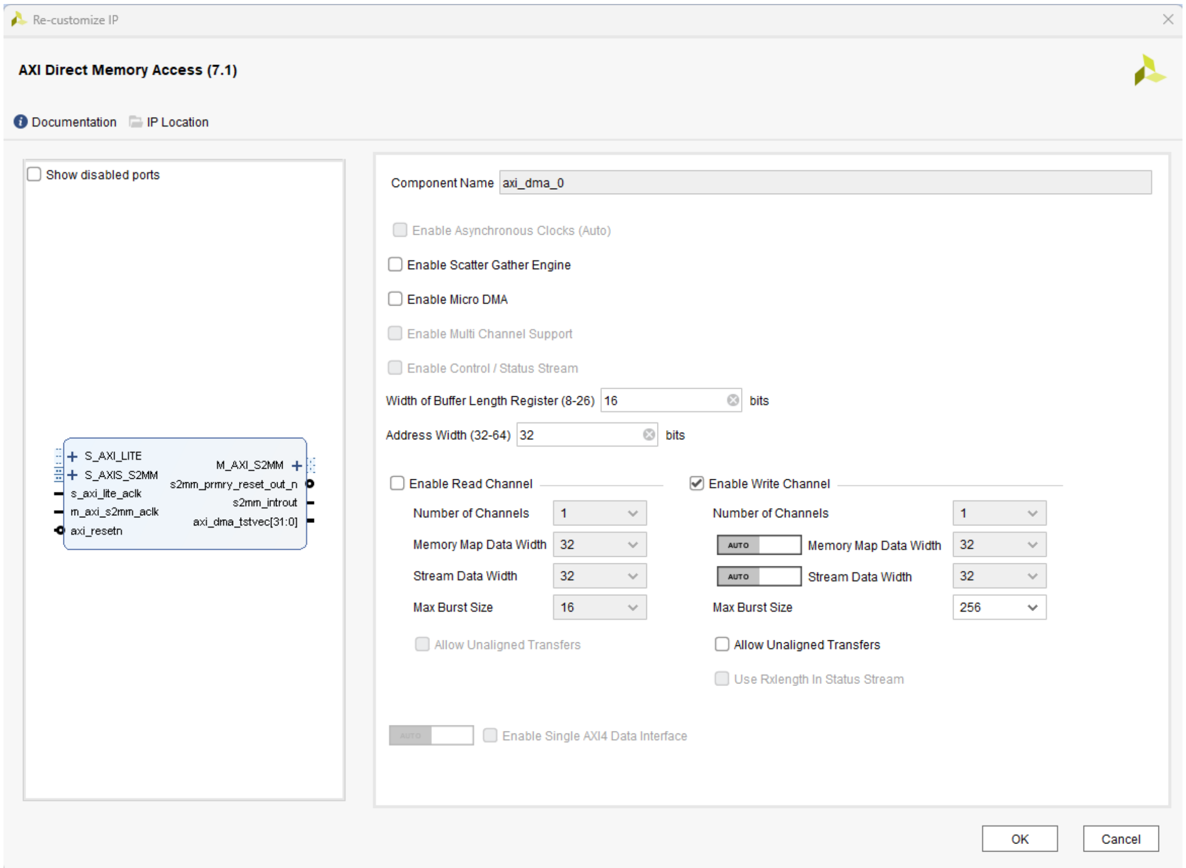
□ Show disabled ports

Component Name xlconstant_0

Const Width 7 [1 - 4096]

Const Val 17

AXI DMA:



AXI Direct Memory Access (7.1)

□ Show disabled ports

Component Name axi_dma_0

□ Enable Asynchronous Clocks (Auto)

□ Enable Scatter Gather Engine

□ Enable Micro DMA

□ Enable Multi Channel Support

□ Enable Control / Status Stream

Width of Buffer Length Register (8-26) 16 bits

Address Width (32-64) 32 bits

□ Enable Read Channel

- Number of Channels 1
- Memory Map Data Width 32
- Stream Data Width 32
- Max Burst Size 16
- □ Allow Unaligned Transfers

☑ Enable Write Channel

- Number of Channels 1
- AUTO Memory Map Data Width 32
- AUTO Stream Data Width 32
- Max Burst Size 256
- □ Allow Unaligned Transfers
- □ Use Rxlength In Status Stream

AUTO □ Enable Single AXI4 Data Interface

```
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;
```

```vhdl
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;
```

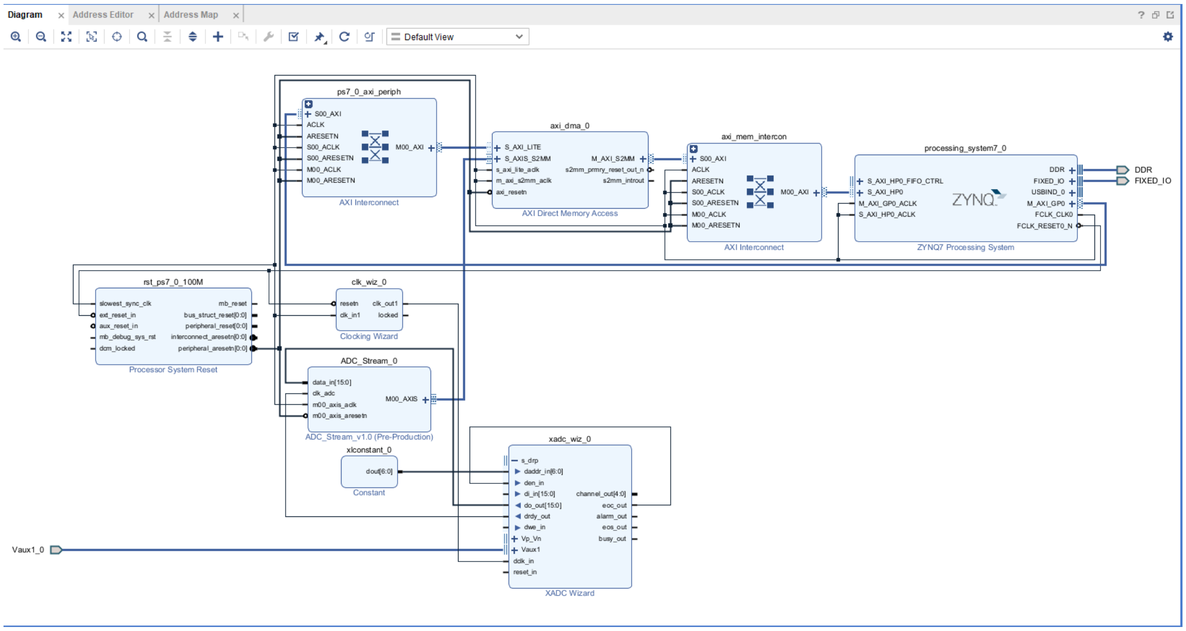


Diagram
Address Editor
Address Map
Default View
ps7_0_axi_periph
S00_AXI
ACLK
ARESETN
S00_ACLK
S00_ARESETN
M00_ACLK
M00_ARESETN
M00_AXI
AXI Interconnect
axi_dma_0
S_AXI_LITE
S_AXIS_S2MM
s_axi_lite_aclk
m_axi_s2mm_aclk
axi_resetn
M_AXI_S2MM
s2mm_prmry_reset_out_n
s2mm_introut
AXI Direct Memory Access
axi_mem_intercon
S00_AXI
ACLK
ARESETN
S00_ACLK
S00_ARESETN
M00_ACLK
M00_ARESETN
M00_AXI
AXI Interconnect
processing_system7_0
S_AXI_HP0_FIFO_CTRL
S_AXI_HP0
M_AXI_GP0_ACLK
S_AXI_HP0_ACLK
ZYNQ
DDR
FIXED_IO
USBIND_0
M_AXI_GP0
FCLK_CLK0
FCLK_RESET0_N
ZYNQ7 Processing System
DDR
FIXED_IO
rst_ps7_0_100M
slowest_sync_clk
ext_reset_in
aux_reset_in
mb_debug_sys_rst
dcm_locked
mb_reset
bus_struct_reset[0:0]
peripheral_reset[0:0]
interconnect_aresetn[0:0]
peripheral_aresetn[0:0]
Processor System Reset
clk_wiz_0
resetn
clk_in1
clk_out1
locked
Clocking Wizard
ADC_Stream_0
data_in[15:0]
clk_adc
m00_axis_aclk
m00_axis_aresetn
M00_AXIS
ADC_Stream_v1.0 (Pre-Production)
xlconstant_0
dout[6:0]
Constant
xadc_wiz_0
s_drp
daddr_in[6:0]
den_in
di_in[15:0]
do_out[15:0]
drdy_out
dwe_in
Vp_Vn
Vaux1
dclk_in
reset_in
channel_out[4:0]
eoc_out
alarm_out
eos_out
busy_out
XADC Wizard
Vaux1_0

```
## 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
```

In [1]:

```
from pynq import PL
from pynq import Overlay
from pynq import allocate

import numpy as np
import matplotlib.pyplot as plt
```

```
Matplotlib is building the font cache; this may take a moment.
```

In [2]:

```
PL.reset()
xadc_stream = Overlay('xstream.bit')

dma       = xadc_stream.axi_dma_0
```

In [3]:

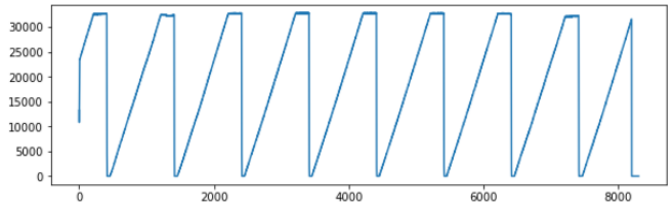
```
buff_deep = 8300

input_buffer = allocate(shape=(buff_deep,), dtype=np.uint32)
```

In [17]:

```
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
```

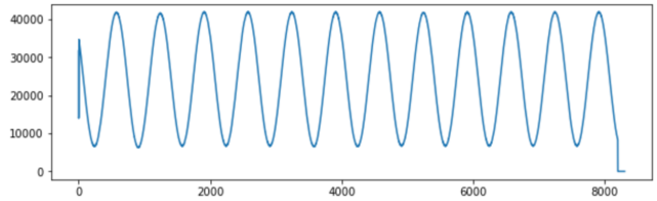
In [18]:

```
plt.figure(figsize=(10,3))
plt.plot(range(0, buff_deep), input_buffer1)
#plt.xlim(0, 400)
plt.show()
```



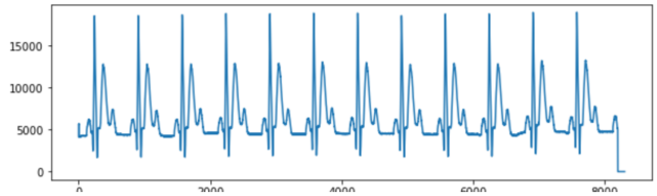
In [22]:

```
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()
```



In [29]:

```
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()
```



```
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()
```