

Figure 2 Output data from FPGA

Figure1 Received Data from FPGA to FX2 Board.



Figure 3 Expected Received Data from FPGA to FX2 Board.

The code :-

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

--library UNISIM;

--use UNISIM.vcomponents.all;

entity fx2lp\_slaveFIFO2b\_streamIN\_fpga\_top is

 Port (

 reset\_n\_out : out STD\_LOGIC; --used for TB

 reset\_in\_n : in std\_LOGIC;

 fdata : inout STD\_LOGIC\_VECTOR(15 downto 0); -- FIFO data lines.

 faddr : out STD\_LOGIC\_VECTOR(1 downto 0); -- FIFO select lines

 slrd : out STD\_LOGIC; -- Read control line

 slwr : out STD\_LOGIC; -- Write control line

 flagd : in STD\_LOGIC; --EP6 full flag

 flaga : in STD\_LOGIC; --EP2 empty flag

 clk : in STD\_LOGIC; --Interface Clock

 sloe : out STD\_LOGIC; --Slave Output Enable control

 clk\_out : out STD\_LOGIC;

 pkt\_end : out STD\_LOGIC;

 done : out STD\_LOGIC;

 sync : inout std\_logic

 );

end fx2lp\_slaveFIFO2b\_streamIN\_fpga\_top;

architecture fx2lp\_slaveFIFO2b\_streamIN\_fpga\_top\_arch of fx2lp\_slaveFIFO2b\_streamIN\_fpga\_top is

component pll\_fx2lp is

 port(

 rst : in std\_logic;

 outclk\_0 : out std\_logic;

 refclk : in std\_logic);

end component pll\_fx2lp;

component ddr

 port(

 datain\_h : in std\_logic;

 datain\_l : in std\_logic;

 outclock : in std\_logic;

 dataout : out std\_logic);

end component;

--streamIN fsm signal

type stream\_in\_state is (stram\_in\_idle, stream\_in\_write);

signal current\_stream\_in\_state, next\_stream\_in\_state : stream\_in\_state;

signal slrd\_n, slwr\_n, sloe\_n,slrd\_d\_n : std\_logic;

--signal CLK\_OUT\_0, clk\_out\_90, clk\_out\_180, CLK\_OUT\_270 : std\_logic;

signal clk\_100 : std\_LOGIC;

signal reset\_n : std\_logic;

signal lock : std\_logic;

signal data\_out : std\_logic\_vector(15 downto 0);

signal done\_d : std\_logic;

signal wait\_s : std\_logic\_vector(3 downto 0);

begin --architecture begining

--pll instantiation

pll\_fx2lp\_inst : component pll\_fx2lp

 port map(

 rst => not reset\_in\_n,

 outclk\_0 => clk\_100,

 refclk => clk

 );

ddr\_inst : ddr

 port map(

 datain\_h => '0',

 datain\_l => '1',

 outclock => clk\_100,

 dataout => clk\_out

 );

--for TB

reset\_n\_out <= reset\_n;

--output signal asignment

sync <= '1';

reset\_n <= sync;

slwr <= slwr\_n;

slrd <= slrd\_n;

sloe <= sloe\_n;

faddr <= "10";

pkt\_end <= '1';

done <= done\_d;

fdata <= data\_out;

process(clk\_100, reset\_n) begin

 if(reset\_n = '0')then

 done\_d <= '0';

 elsif(clk\_100'event and clk\_100 = '1')then

 if(wait\_s = "1010")then

 done\_d <= '1';

 end if;

 end if;

end process;

process(clk\_100, reset\_n) begin

 if(reset\_n = '0')then

 wait\_s <= "0000";

 elsif(clk\_100'event and clk\_100 = '1')then

 if(wait\_s < "1010")then

 wait\_s <= wait\_s + '1';

 end if;

 end if;

end process;

--write control signal generation

process(current\_stream\_in\_state, flagd)begin

 if((current\_stream\_in\_state = stream\_in\_write) and (flagd = '1'))then

 slwr\_n <= '1';

 else

 slwr\_n <= '0';

 end if;

end process;

--streamIN mode state machine

streamIN\_mode\_fsm\_f : process(clk\_100, reset\_n) begin

 if(reset\_n = '0')then

 current\_stream\_in\_state <= stram\_in\_idle;

 elsif(clk\_100'event and clk\_100 = '1')then

 current\_stream\_in\_state <= next\_stream\_in\_state;

 end if;

end process;

--LoopBack mode state machine combo

streamIN\_mode\_fsm : process(flaga, flagd, current\_stream\_in\_state) begin

 next\_stream\_in\_state <= current\_stream\_in\_state;

 case current\_stream\_in\_state is

 when stram\_in\_idle =>

 if((flagd = '1') and (sync = '1'))then

 next\_stream\_in\_state <= stream\_in\_write;

 else

 next\_stream\_in\_state <= stram\_in\_idle;

 end if;

 when stream\_in\_write =>

 if(flagd = '0')then

 next\_stream\_in\_state <= stram\_in\_idle;

 else

 next\_stream\_in\_state <= stream\_in\_write;

 end if;

 when others =>

 next\_stream\_in\_state <= stram\_in\_idle;

 end case;

end process;

--data generator counter

process(clk\_100, reset\_n) begin

 if(reset\_n = '0')then

 data\_out <= "0000000000000000";

 elsif(clk\_100'event and clk\_100 = '1')then

 if(slwr\_n = '0')then

 data\_out <= data\_out + '1';

 end if;

 end if;

 -- end if;

end process;

end architecture;