The Abdus Salam
International Centre
for Theoretical Physics

United Nations Educational, Scientific and Cultural Organization

IAEA International Atomic Energy Agency

2384-27

ICTP Latin-American Advanced Course on FPGADesign for Scientific Instrumentation

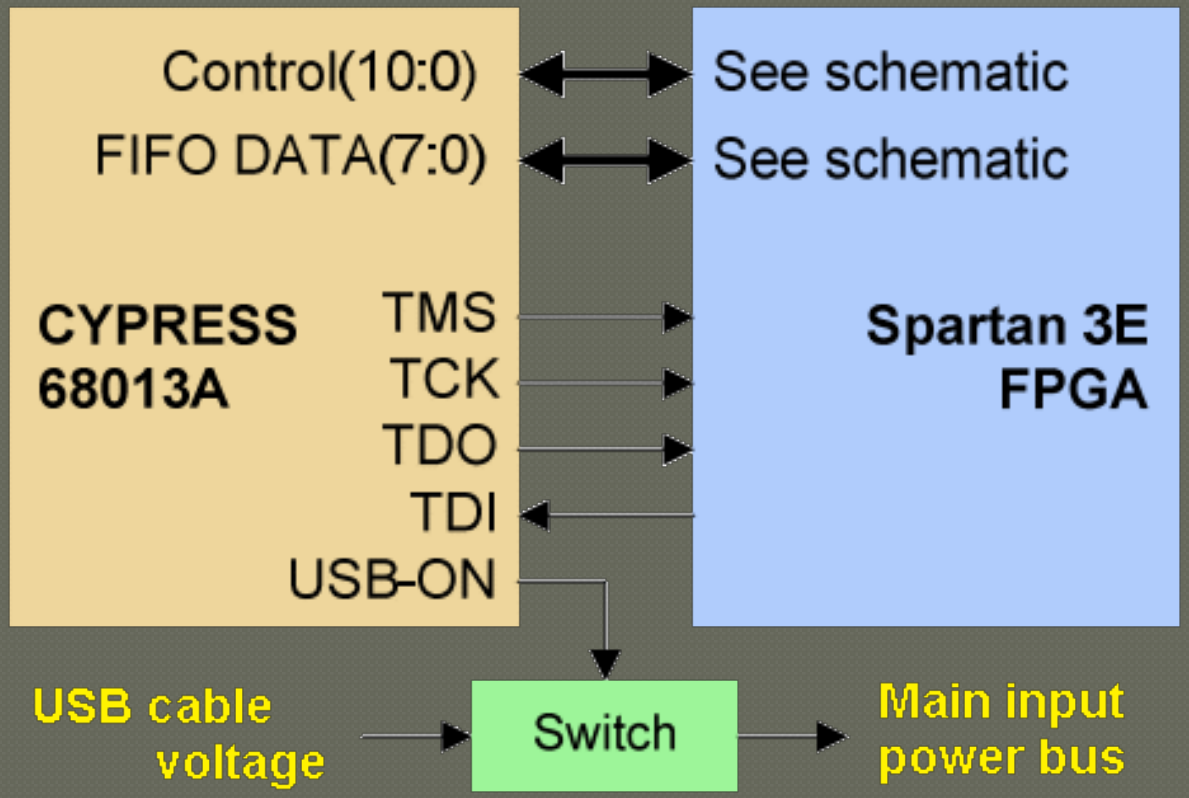
19 November - 7 December, 2012

Sequential Logic Described in VHDL - A Design Example

ARTECHE DIAZ Raul

Center of Applied Studies for Nuclear Development
(CEADEN)
Calle 30# 502, Entre 5Ta Y 7 Ma Ave., Miramar
(PO Box 6122), 11300 Havana
CUBA

Tarjeta Nexys2 - Conexión USB

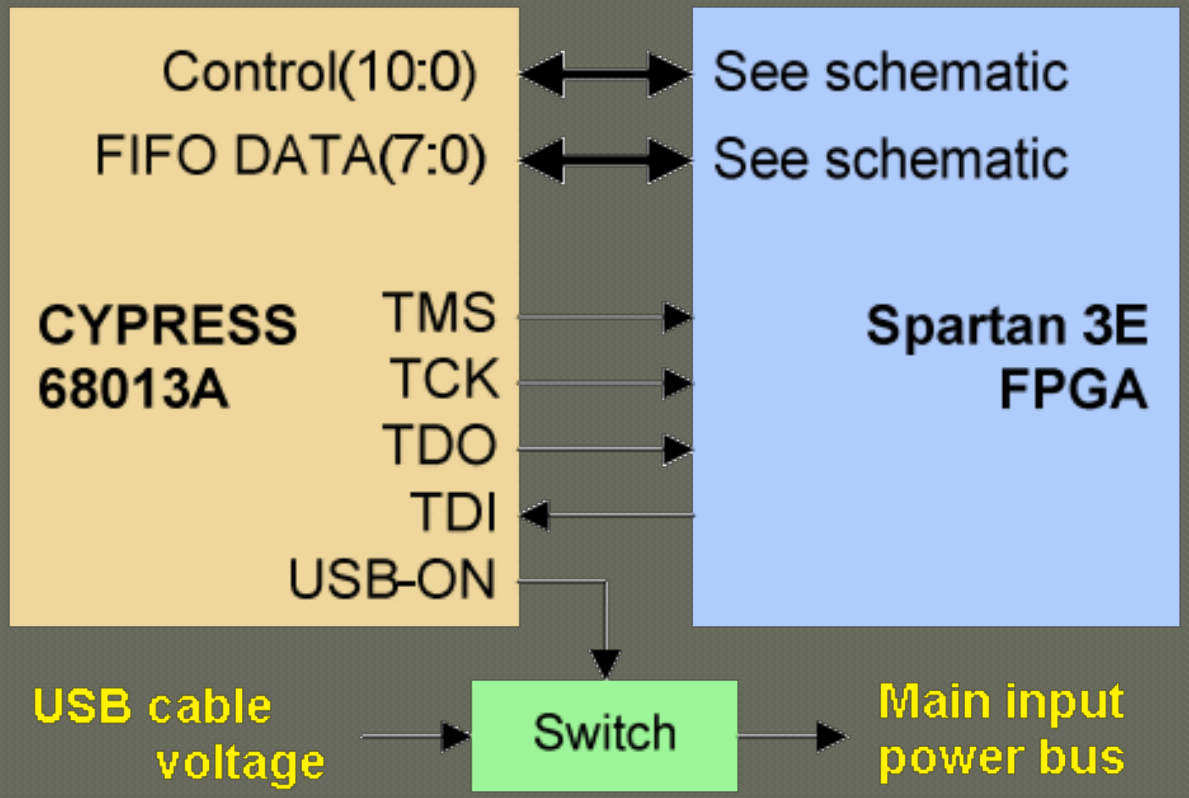


Que puedo hacer a través de la conexión USB ?

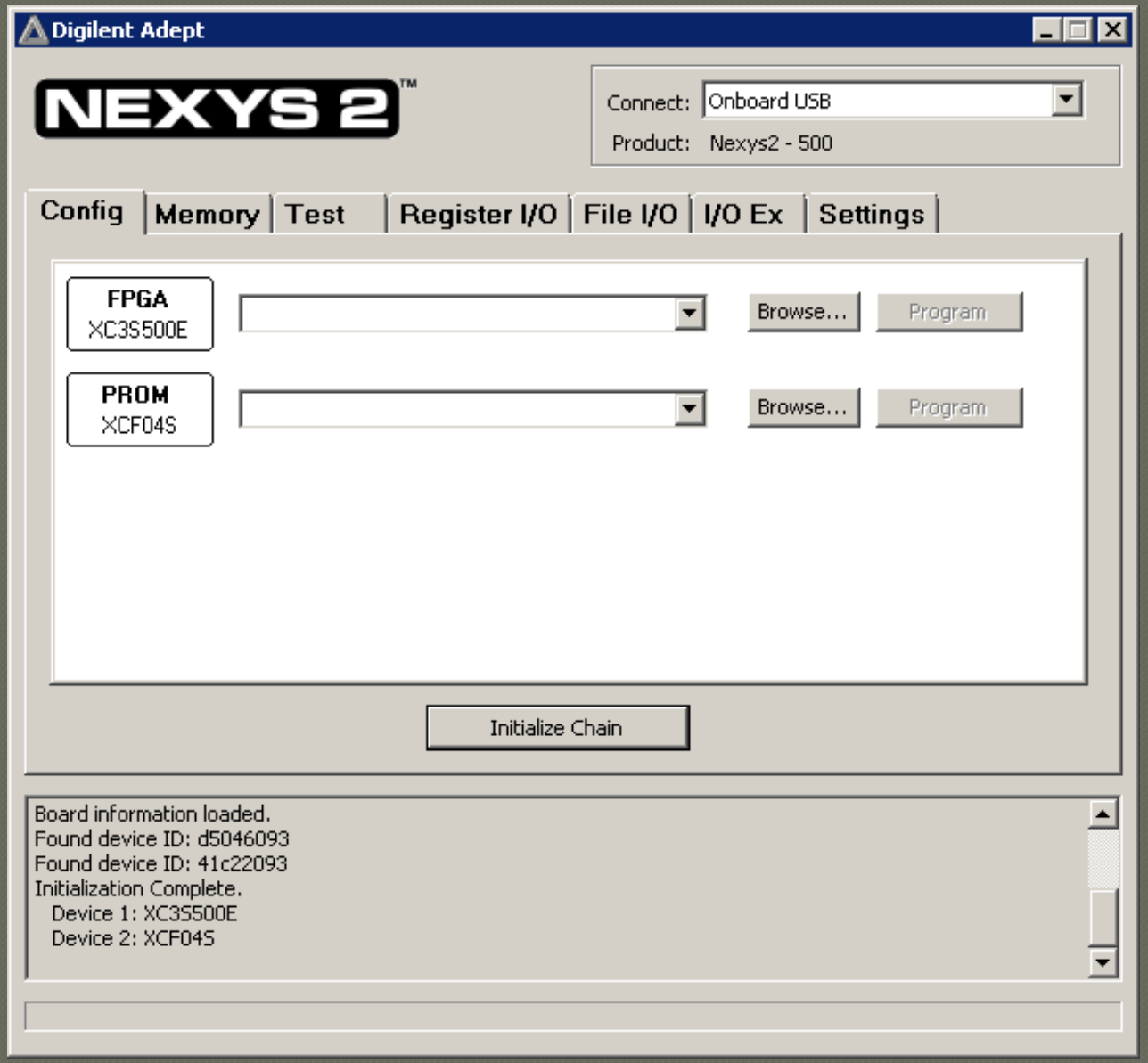
Como transfiero los datos a la FPGA ?

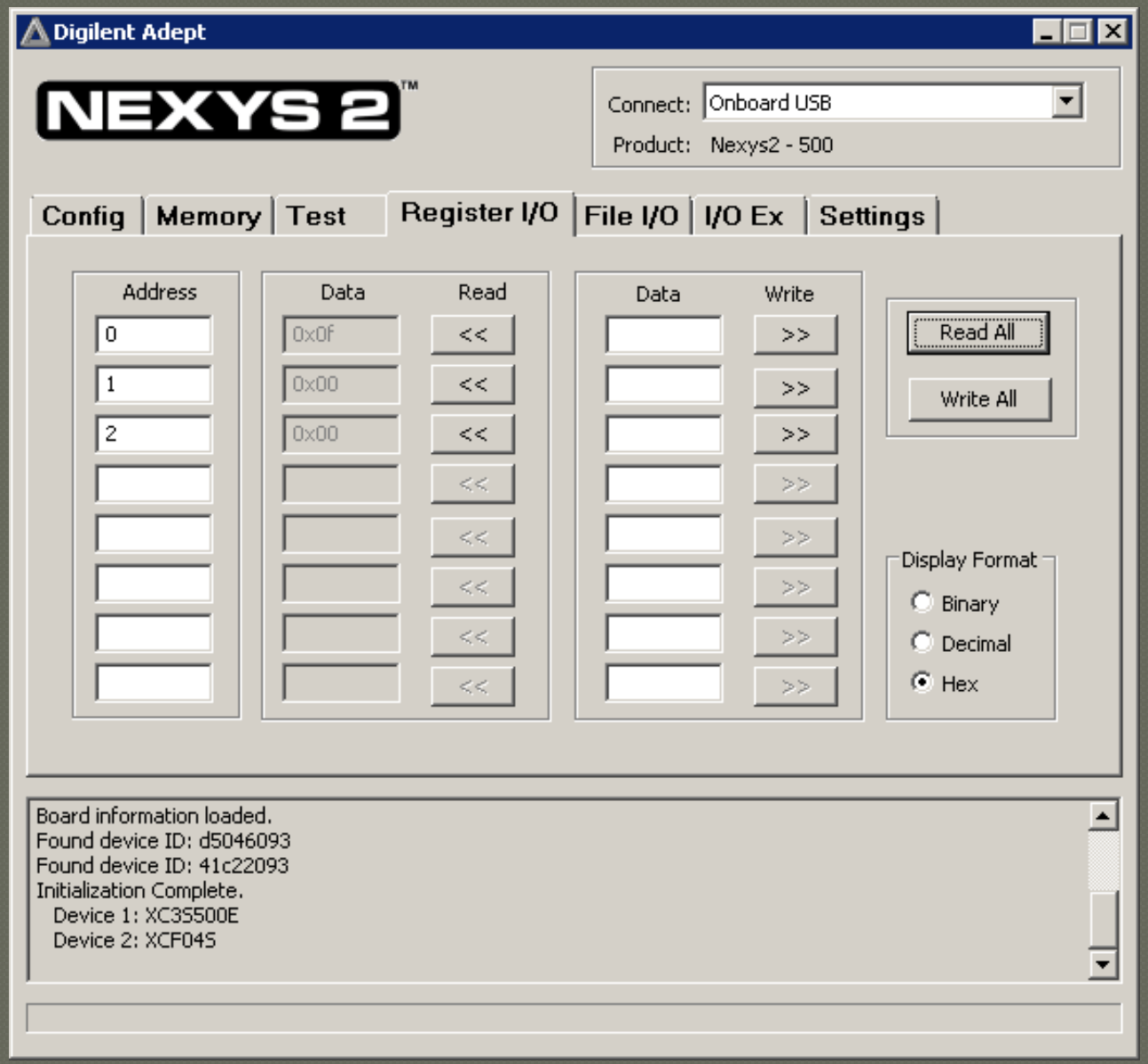
Que lógica debo implementar ?

Tarjeta Nexys2 - Conexión USB



- Alimentar la tarjeta desde la PC (300 mA).
- Programar la FPGA desde el puerto USB.
- Transferir datosa una velocidad máxima de 38 MB/sec.


Digilent Adept
NEXYS 2™
Connect: Onboard USB
Product: Nexys2 - 500
Config
Memory
Test
Register I/O
File I/O
I/O Ex
Settings
FPGA
XC3S500E
Browse...
Program
PROM
XCF04S
Browse...
Program
Initialize Chain
Board information loaded.
Found device ID: d5046093
Found device ID: 41c22093
Initialization Complete.
Device 1: XC3S500E
Device 2: XCF04S



Raúl Arteche Díaz arteche@ceaden.edu.cu Noviembre 2012

DEPP: Digilent Asynchronous Parallel Interface

- La interfaz DEPP consiste en un bus de dato bidireccional de 8-bits y cuatro señales de control.

- La transferencia de datos es asincrónica y se realiza a través de siclos de buses.

DSTM : Digilent Synchronous Parallel Interface

- La interfaz DSTM consiste en un bus de dato bidireccional de 8-bits y nueve señales de control.

- La transferencia de datos es sincrónica y opera basado en un reloj generado por el microcontrolador USB.

Raúl Arteche Díaz arteche@ceaden.edu.cu Noviembre 2012

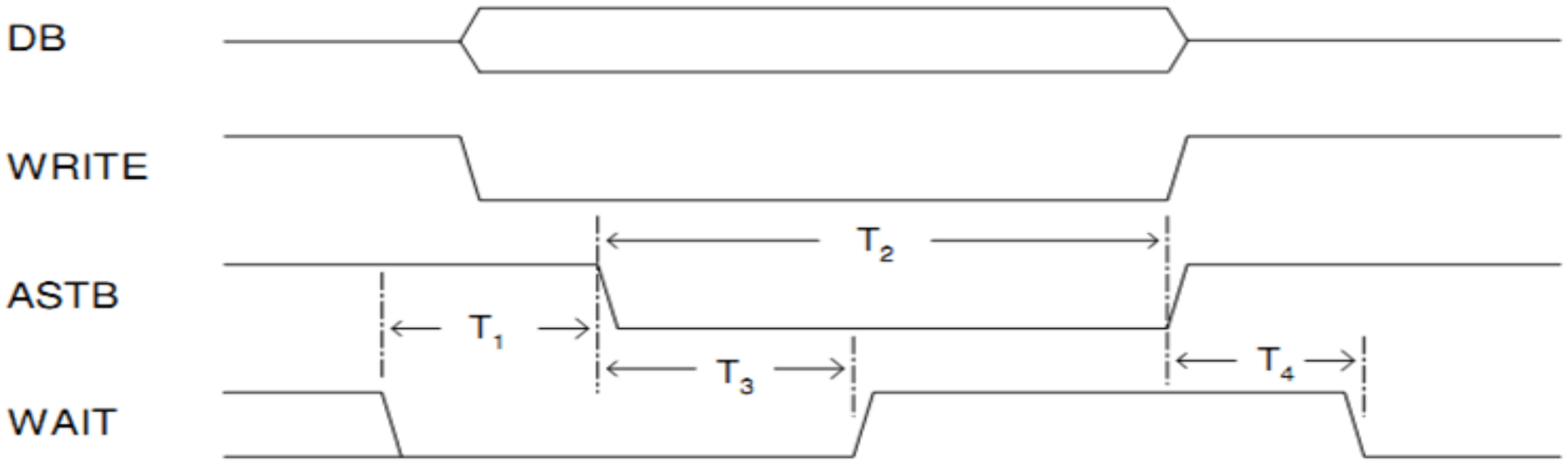
DEPP: Digilent Asynchronous Parallel Interface

Nombre	Fuente	Descripción
DB0 – DB7	bidir	Bus de datos Bidireccional. El host es la fuente durante los ciclos de escritura y el periférico es la fuente durante los ciclos de lectura.
WRITE	host	Controla la dirección de la transferencia. '1' = Lectura, el host lee del periférico. '0' = Escritura, el host escribe en el periférico.
ASTB	host	Address strobe. Escribir en el registro de direcciones.
DSTB	host	Data strobe. Escribir o Leer el registro Datos.
de WAIT aceptar	peripheral	Señal de Handshake usada para indicar cuando el periférico esta listo para datos o tiene datos disponibles.

Timing Diagrams

Parameter	Description	Time
T_1	WAIT low to ASTB or DSTB active	40ns min
T_2	ASTB or DSTB active	80ns min
T_3	ASTB or DSTB active to WAIT high	0ns to 10ms
T_4	ASTB or DSTB inactive to WAIT low	0ns to 10ms
T_5	DSTB low to DB valid	20ns max
T_6	DSTB high to DB invalid	20ns min

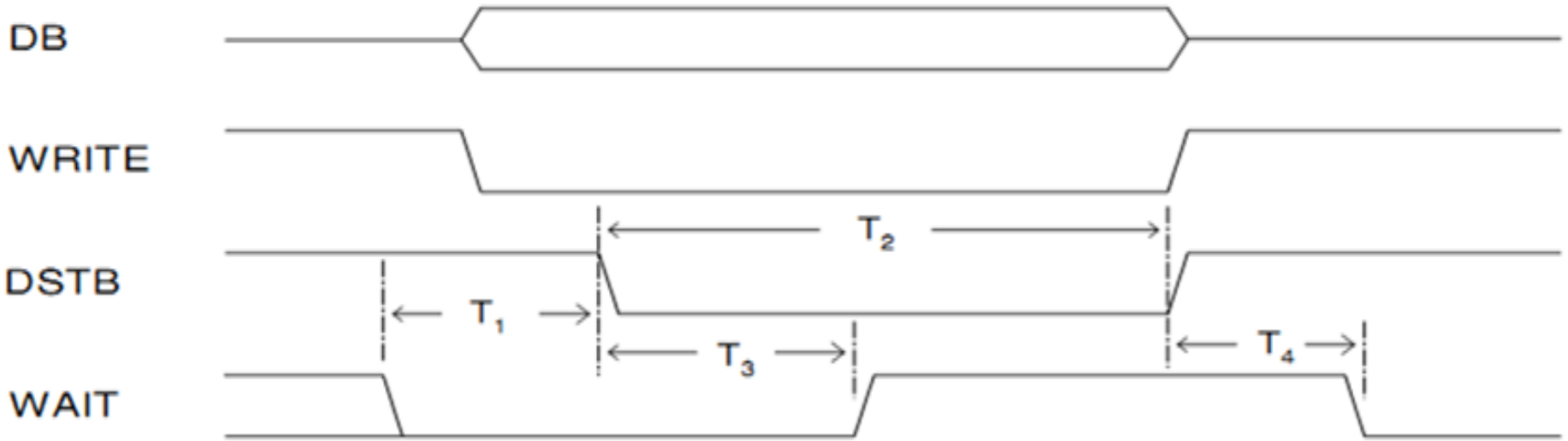
Address Write



Timing Diagrams

Parameter	Description	Time
T_1	WAIT low to ASTB or DSTB active	40ns min
T_2	ASTB or DSTB active	80ns min
T_3	ASTB or DSTB active to WAIT high	0ns to 10ms
T_4	ASTB or DSTB inactive to WAIT low	0ns to 10ms
T_5	DSTB low to DB valid	20ns max
T_6	DSTB high to DB invalid	20ns min

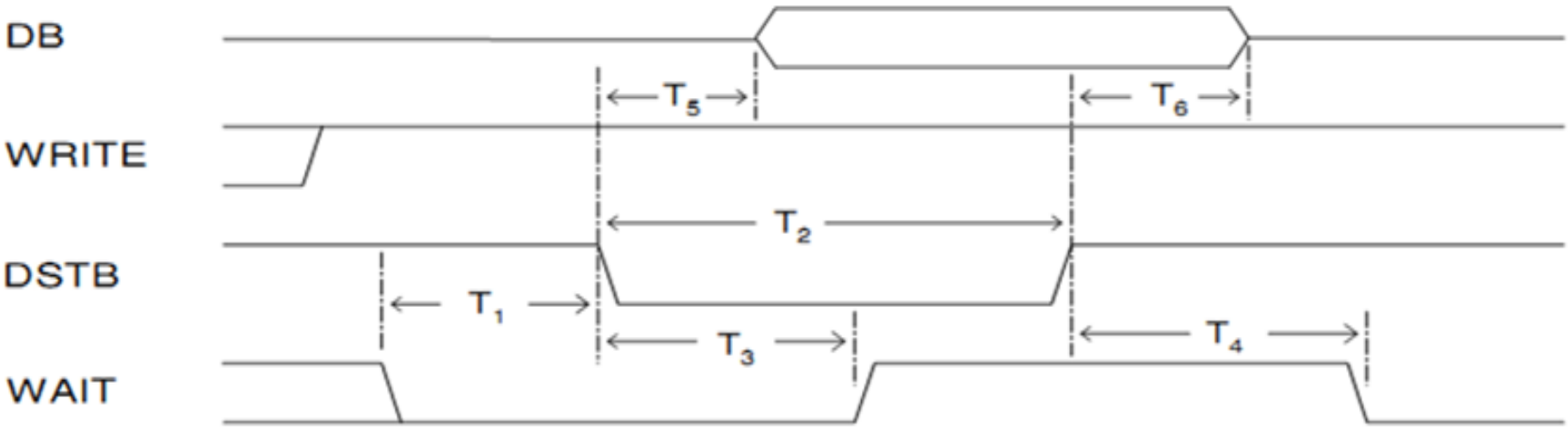
Data Write



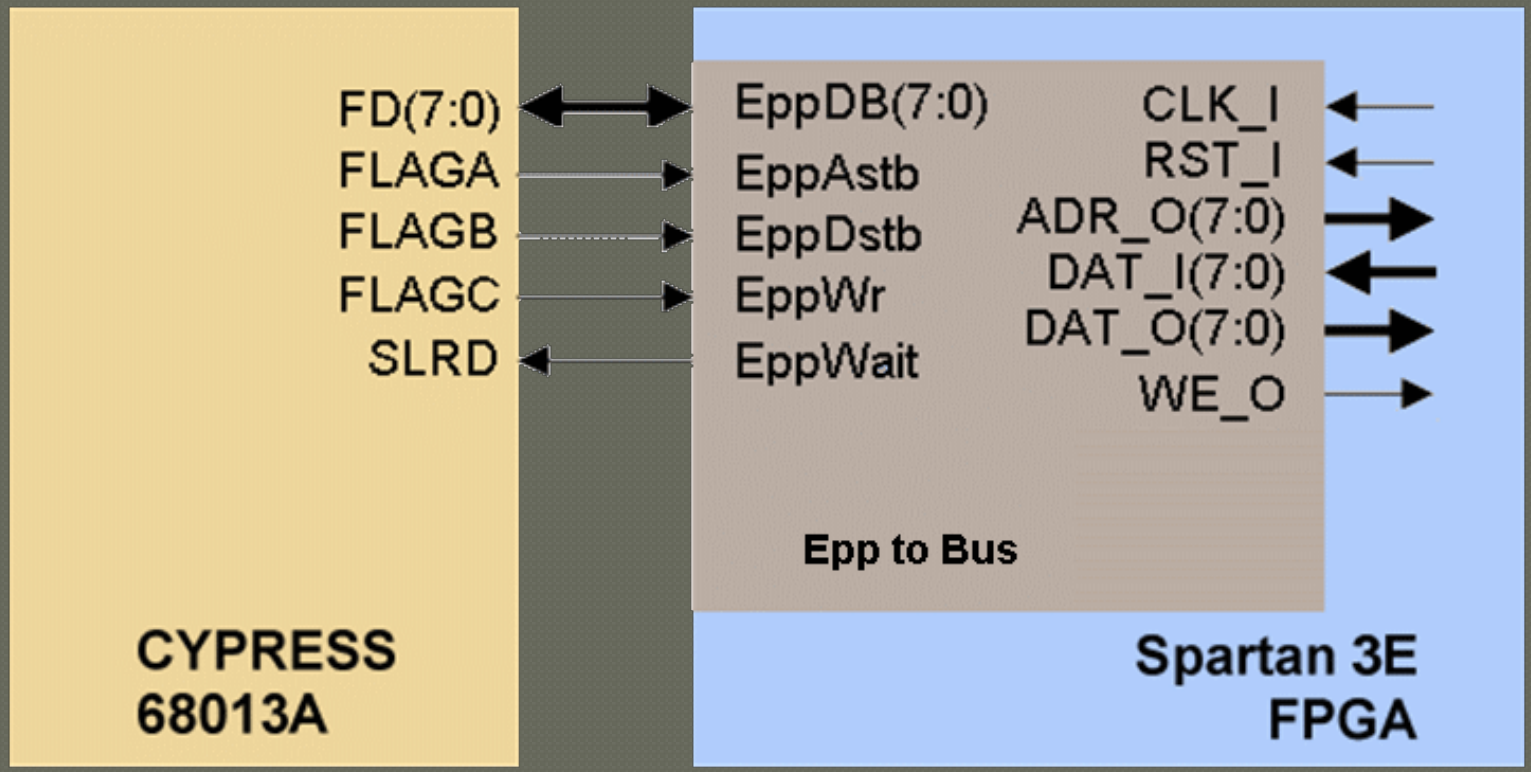
Timing Diagrams

Parameter	Description	Time
T_1	WAIT low to ASTB or DSTB active	40ns min
T_2	ASTB or DSTB active	80ns min
T_3	ASTB or DSTB active to WAIT high	0ns to 10ms
T_4	ASTB or DSTB inactive to WAIT low	0ns to 10ms
T_5	DSTB low to DB valid	20ns max
T_6	DSTB high to DB invalid	20ns min

Data Read

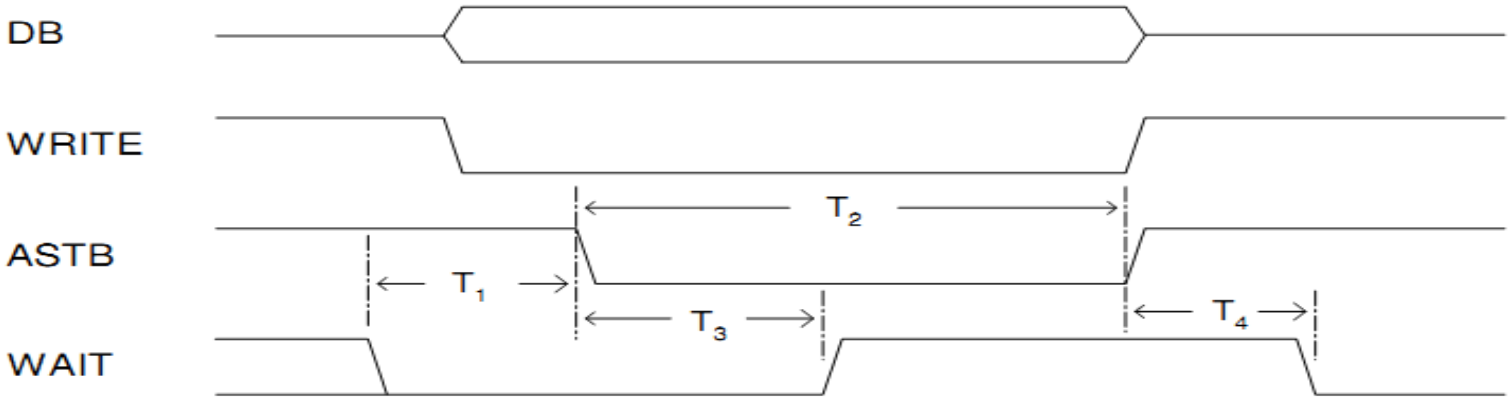


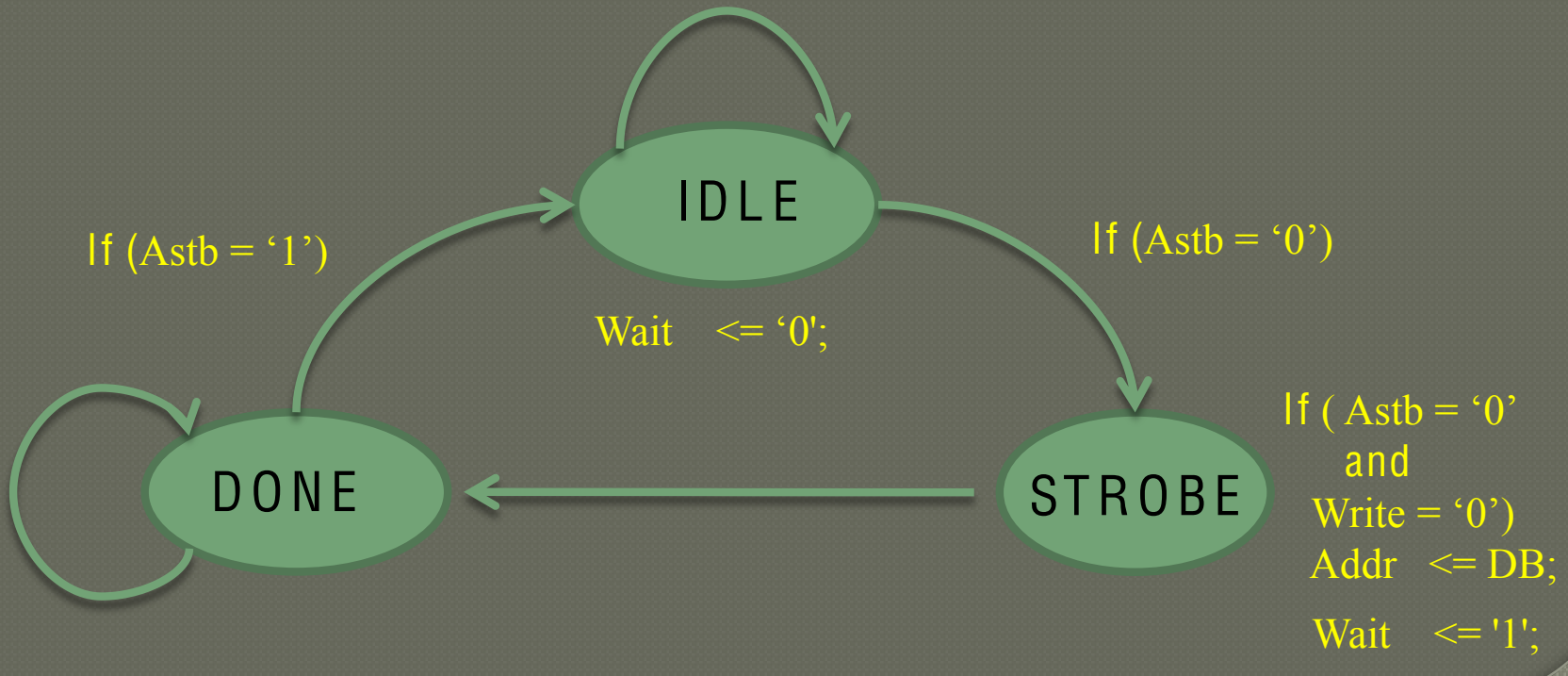
DEPP: Digilent Asynchronous Parallel Interface

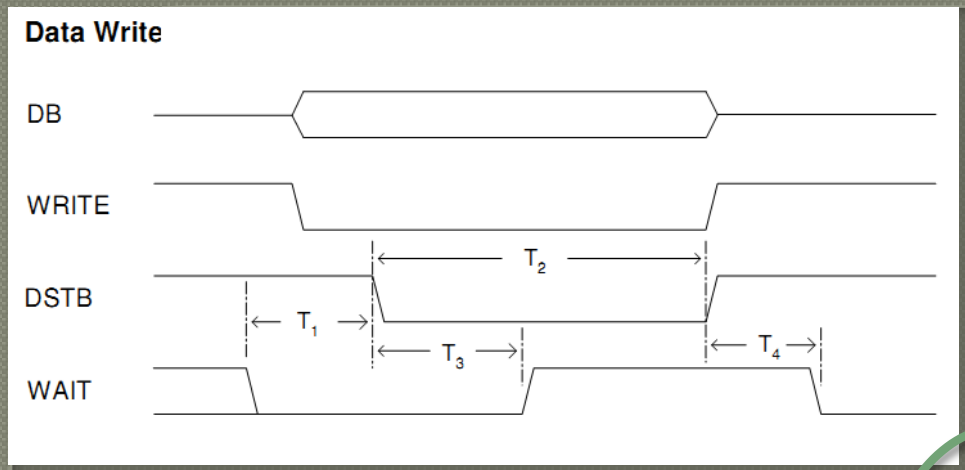


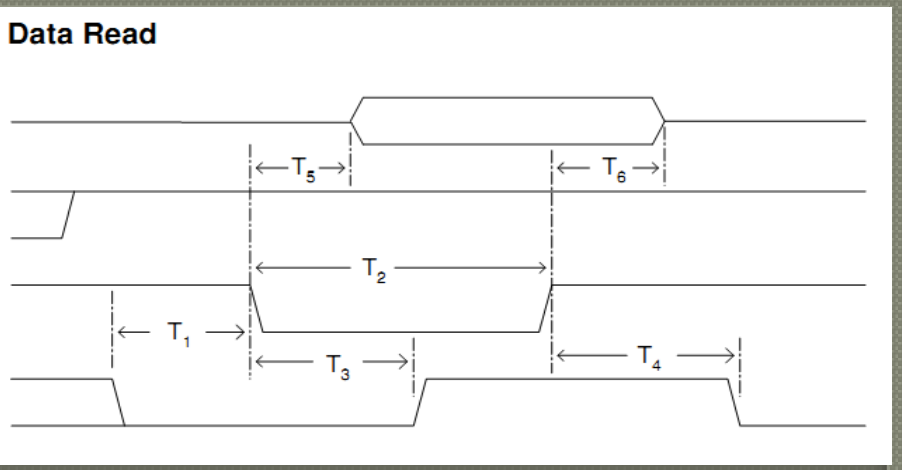
Raúl Arteche Díaz arteche@ceaden.edu.cu Noviembre 2012

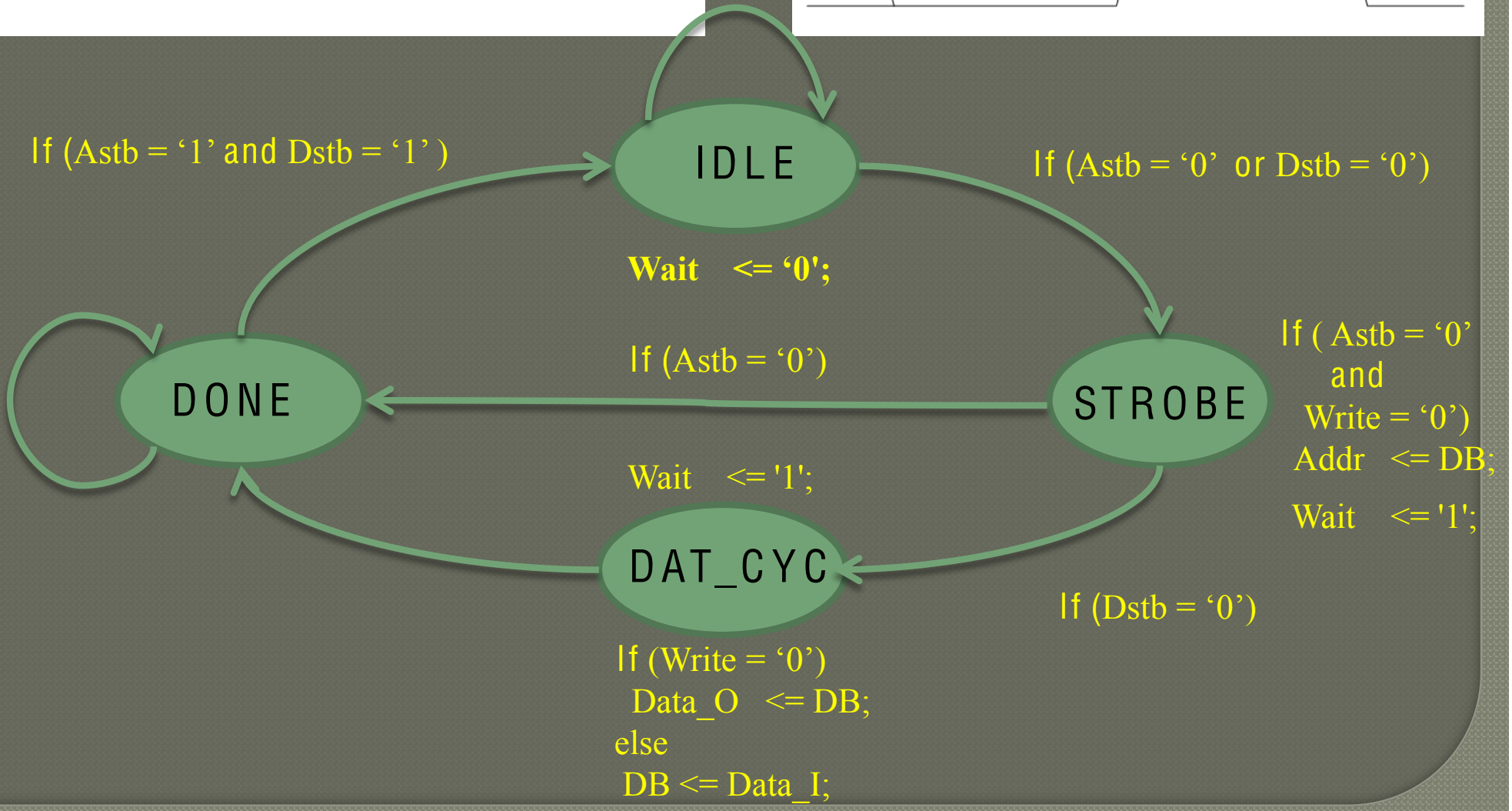
Address Write

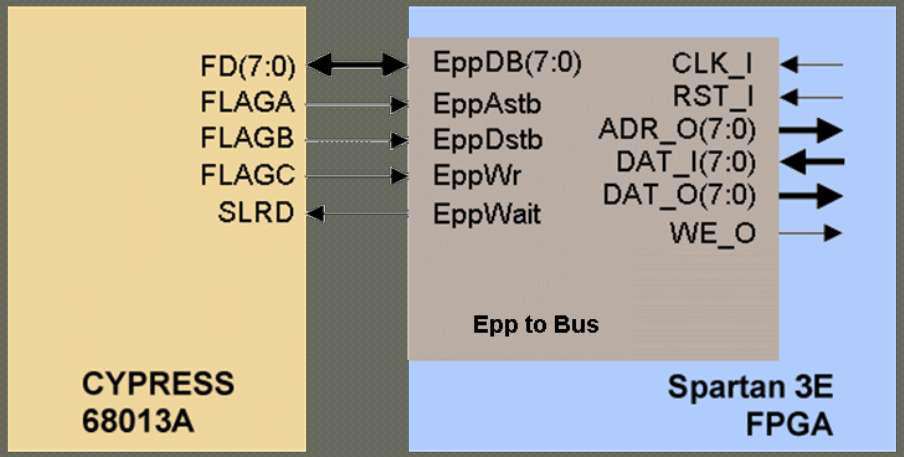




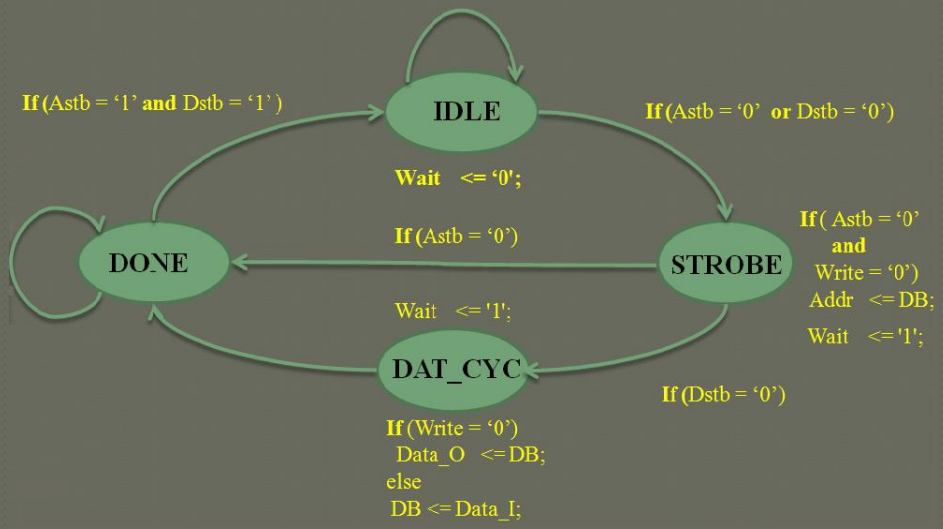

Data Write
DB
WRITE
DSTB
WAIT
T1
T2
T3
T4


Data Read
T1
T2
T3
T4
T5
T6


If (Astb = '1' and Dstb = '1')
IDLE
If (Astb = '0' or Dstb = '0')
Wait <= '0';
If (Astb = '0')
DONE
STROBE
If (Astb = '0' and Write = '0')
Addr <= DB;
Wait <= '1';
Wait <= '1';
DAT_CYC
If (Dstb = '0')
If (Write = '0')
Data_O <= DB;
else
DB <= Data_I;

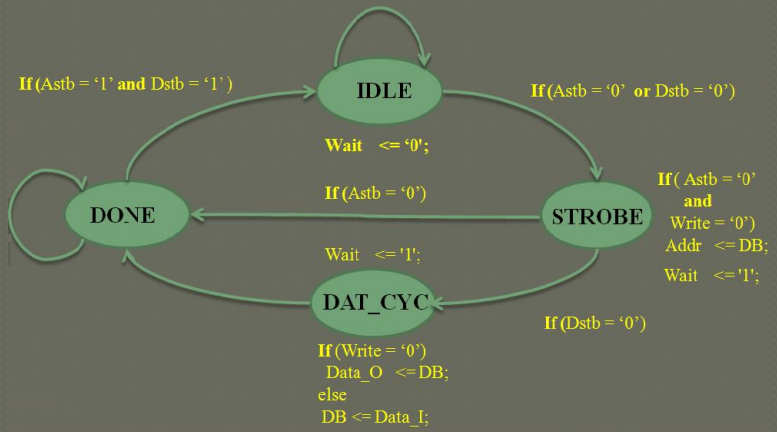



```
entity EppToBus is
  port (
        CLK_I     : in    std_logic;
        RST_I     : in    std_logic;

        EppDB     : inout  std_logic_vector(7 downto 0);
        EppWr     : in     std_logic;
        EppAstb   : in     std_logic;

        EppDstb   : in     std_logic;
        EppWait   : out    std_logic;

        ADR_O     : out   std_logic_vector(7 downto 0);

        DAT_I     : in    std_logic_vector(7 downto 0);
        DAT_O     : out   std_logic_vector(7 downto 0);

        WE_       : out   std_logic
  );
end EppToBus
```




```
type   STATE_TYPE is (EPP_IDLE, EPP_STROBE, EPP_DONE, EPP_DAT_CYC);

signal EPP_STATE : STATE_TYPE;
```

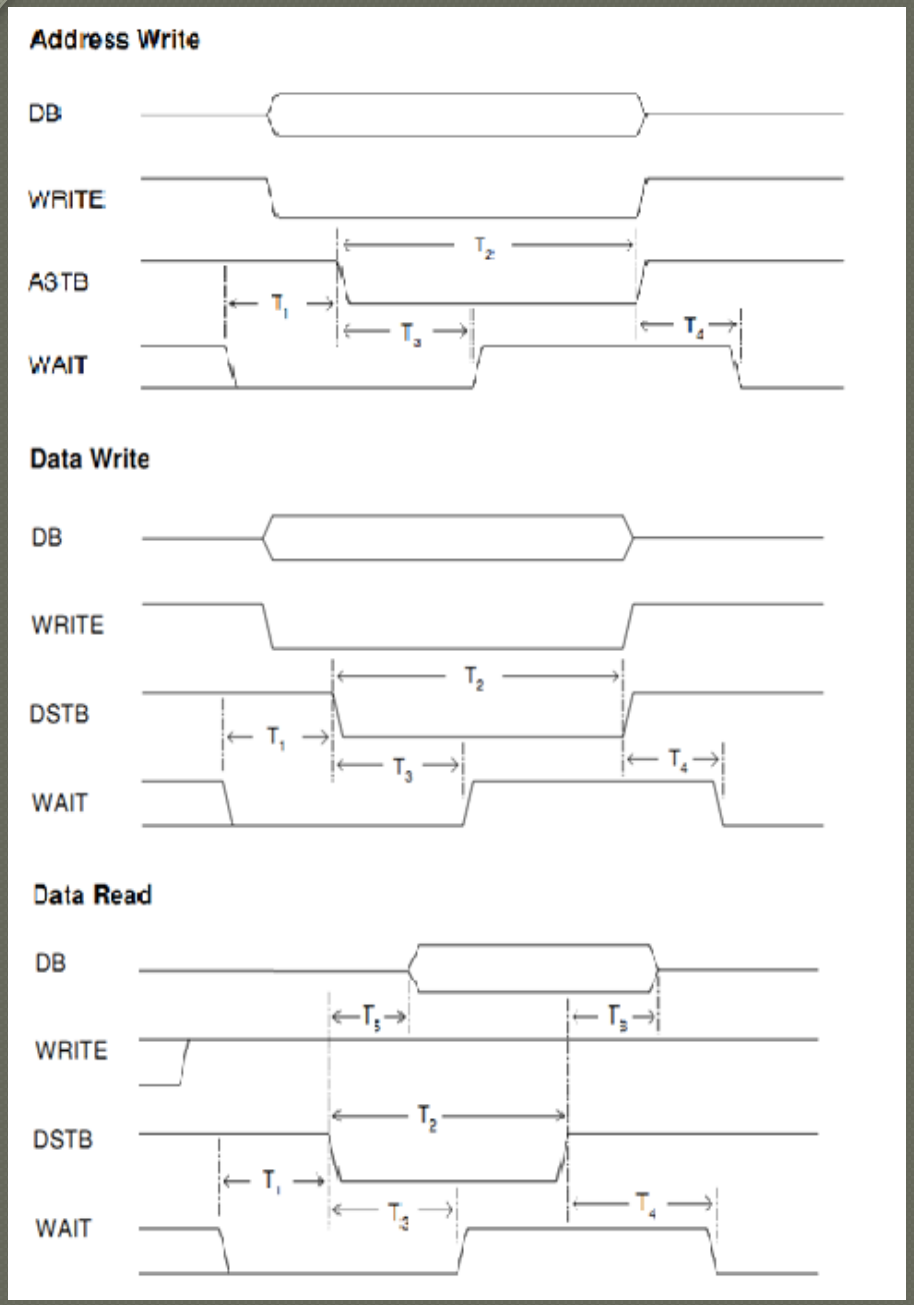
```
EppDB  <= sEppOut  when EppWr = '1' else (others => 'Z');

DAT_O        <= sEppIn;
ADR_O        <= sEppAddr;

process(CLK_I,RST_I)
begin
    if (RST_I = '1') then
      EPP_STATE   <= EPP_IDLE;
      EppWait   <= '0';
      sEppOut   <= (others => '0');
      WE_O      <= '0';

   elsif rising_edge(CLK_I) then
       case EPP_STATE is
           when EPP_IDLE =>
                if (EppDstb = '0') or (EppAstb = '0') then
                   EPP_STATE <= EPP_STROBE;
                else
                   EPP_STATE <= EPP_IDLE;
                end if;

                EppWait   <= '0';

           when EPP_STROBE =>
                if (EppAstb = '0') then
                  if (EppWr = '0') then
                     EPP_STATE  <= EPP_DONE;
                     sEppAddr   <= EppDB;
                     EppWait    <= '1';
                  end if;
                elsif (EppDstb = '0') then
                  EPP_STATE <= EPP_DAT_CYC;
                end if;
```



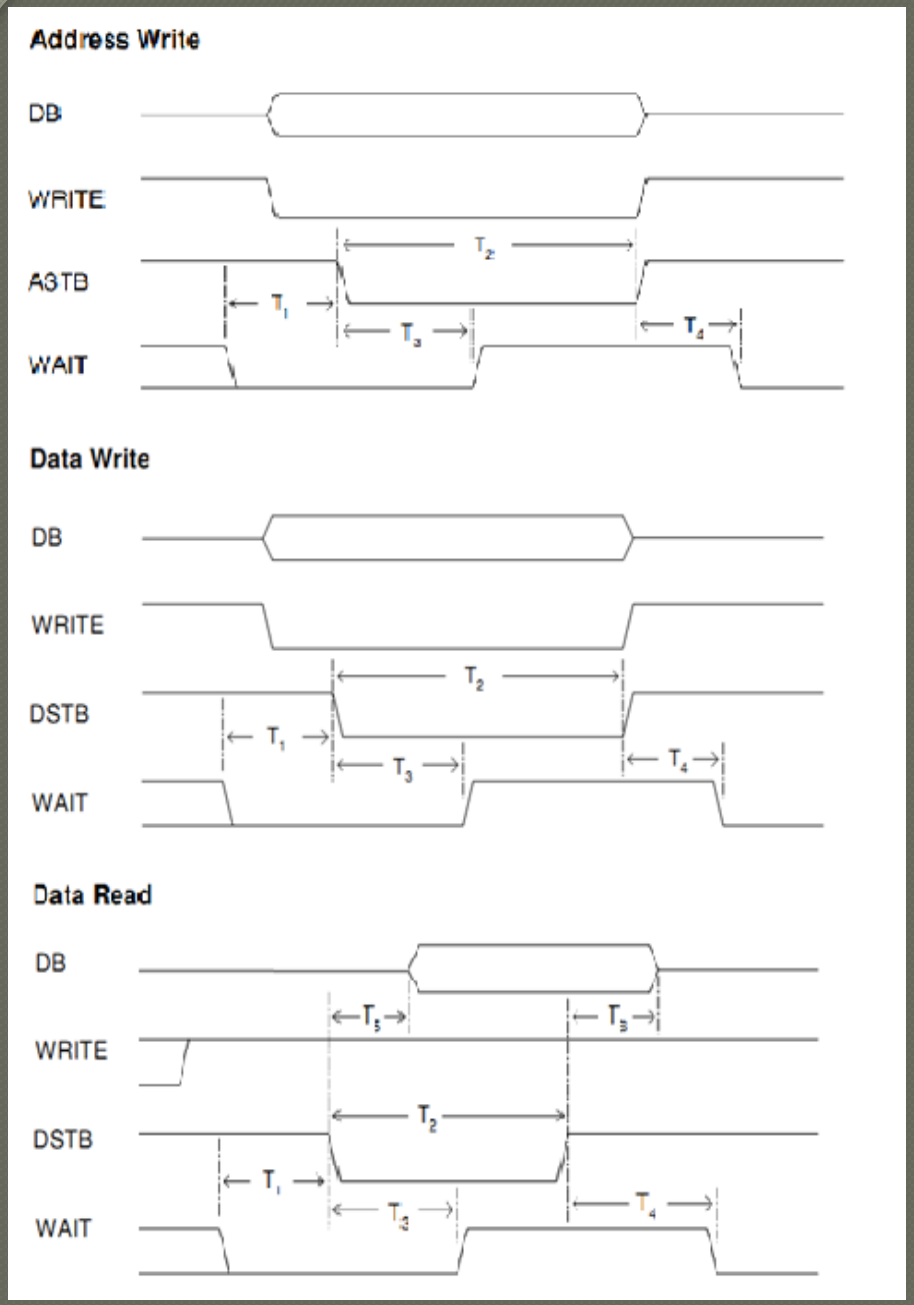

```
           when EPP_DAT_CYC =>
                EPP_STATE <= EPP_DONE;
                if (EppWr = '0') then
                   sEppIn   <= EppDB;
                   WE_O     <= '1';
                else
                   sEppOut  <= DAT_I;
                end if;

                EppWait    <= '1';

           when EPP_DONE =>
                if (EppDstb = '1') and (EppAstb = '1') then
                   EPP_STATE <= EPP_IDLE;
                   WE_O         <= '0';
                else
                   EPP_STATE <= EPP_DONE;
                end if;
           when others =>
                EPP_STATE <= EPP_IDLE;
       end case;
    end if;

end process;
```

Como compruebo que mi código VHDL funciona?

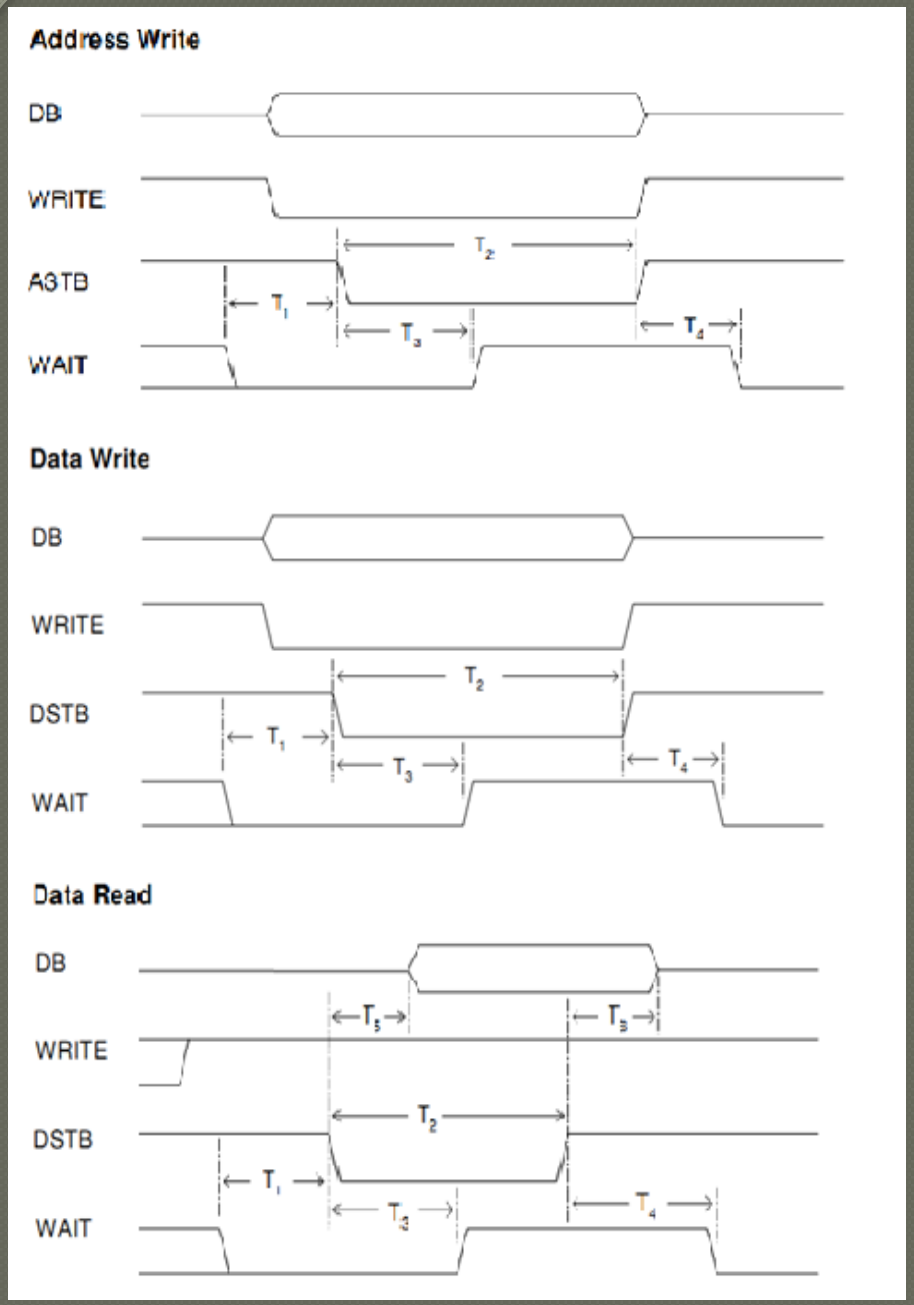
Raúl Arteche Díaz arteche@ceaden.edu.cu Noviembre 2012




```
procedure EppWriteAddress(
                Address : in  std_logic_vector(7 downto 0);
                signal EppWait : in    std_logic;
                signal EppDB   : out   std_logic_vector(7 downto
0);
                signal EppAstb : out   std_logic;
                signal EppWr   : out   std_logic)  is

                constant T1 : Time :=  41 ns;
                -- WAIT low to ASTB or DSTB active min time
                constant T2 : Time :=  81 ns;
                -- ASTB or DSTB active min time

begin

                EppDB       <= Address;
                EppWr       <= '0';
                wait for 36 ns;
                EppAstb     <= '0';

                report "Waiting for rising edge of EppWait in
                                EppWriteAddrees Procedure";
                wait until      EppWait ='1';

                wait for T2;

                EppAstb     <= '1';
                EppWr       <= '1';

                wait for T1;
                 -- WAIT low to ASTB or DSTB active 40ns mini

end EppWriteAddress;
```



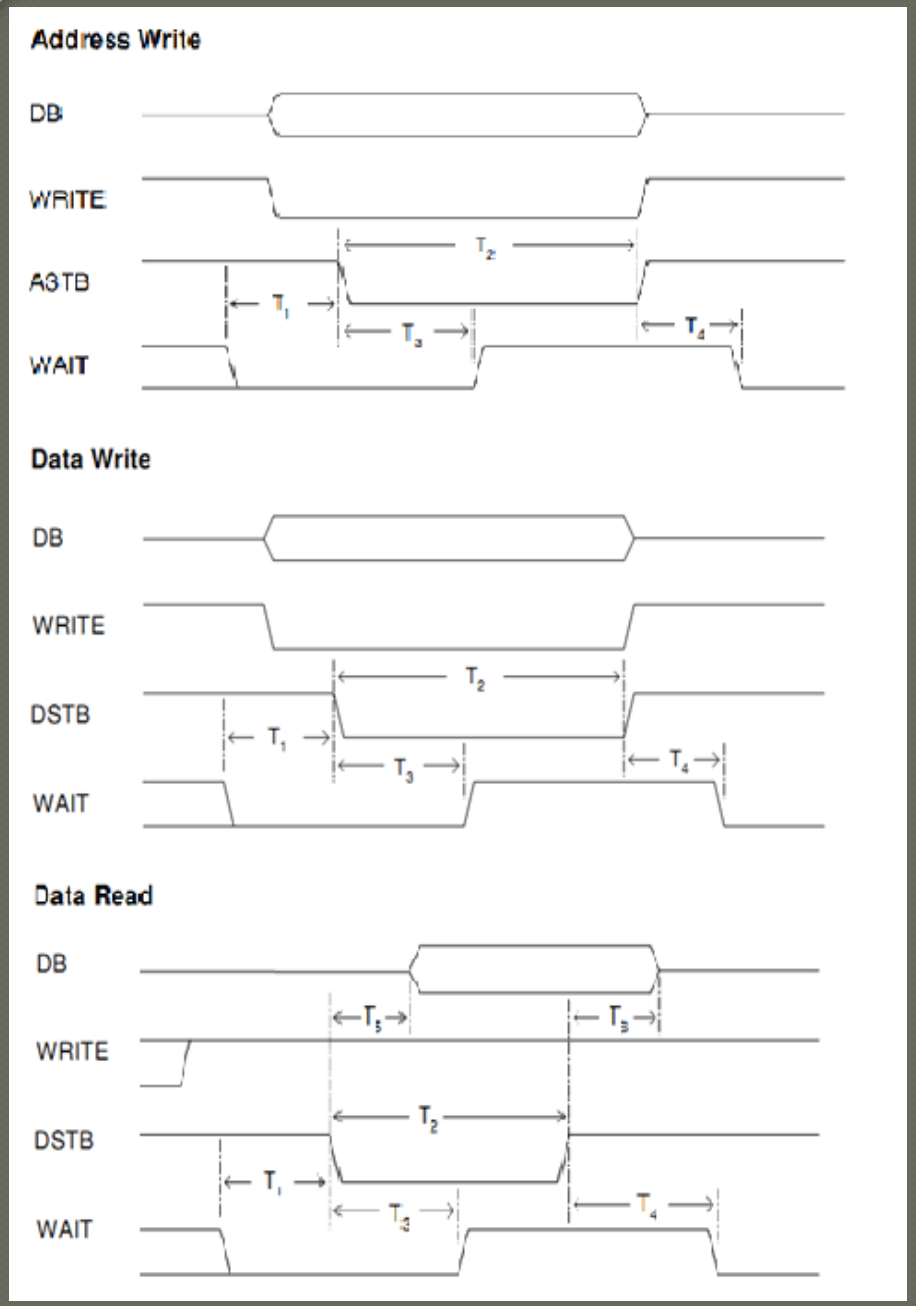

```
procedure EppWriteData(
                Data : in       std_logic_vector(7 downto 0);
                signal EppWait : in     std_logic;
                signal EppDB   : out   std_logic_vector(7 downto
0);
                signal EppDstb : out   std_logic;
                signal EppWr    : out   std_logic)  is

                constant T1 : Time :=  41 ns;
                -- WAIT low to ASTB or DSTB active min time
                constant T2 : Time :=  81 ns;
                -- ASTB or DSTB active min time

begin

                EppDB         <= Data;
                EppWr         <= '0';
                wait for 36 ns;
                EppDstb       <= '0';

                report "Waiting for rising edge of EppWait in
                                EppWriteData Procedure";
                wait until      EppWait ='1';

                wait for T2;

                EppDstb       <= '1';
                EppWr         <= '1';

                wait for T1;
                 -- WAIT low to ASTB or DSTB active 40ns mini

end EppWriteData;
```

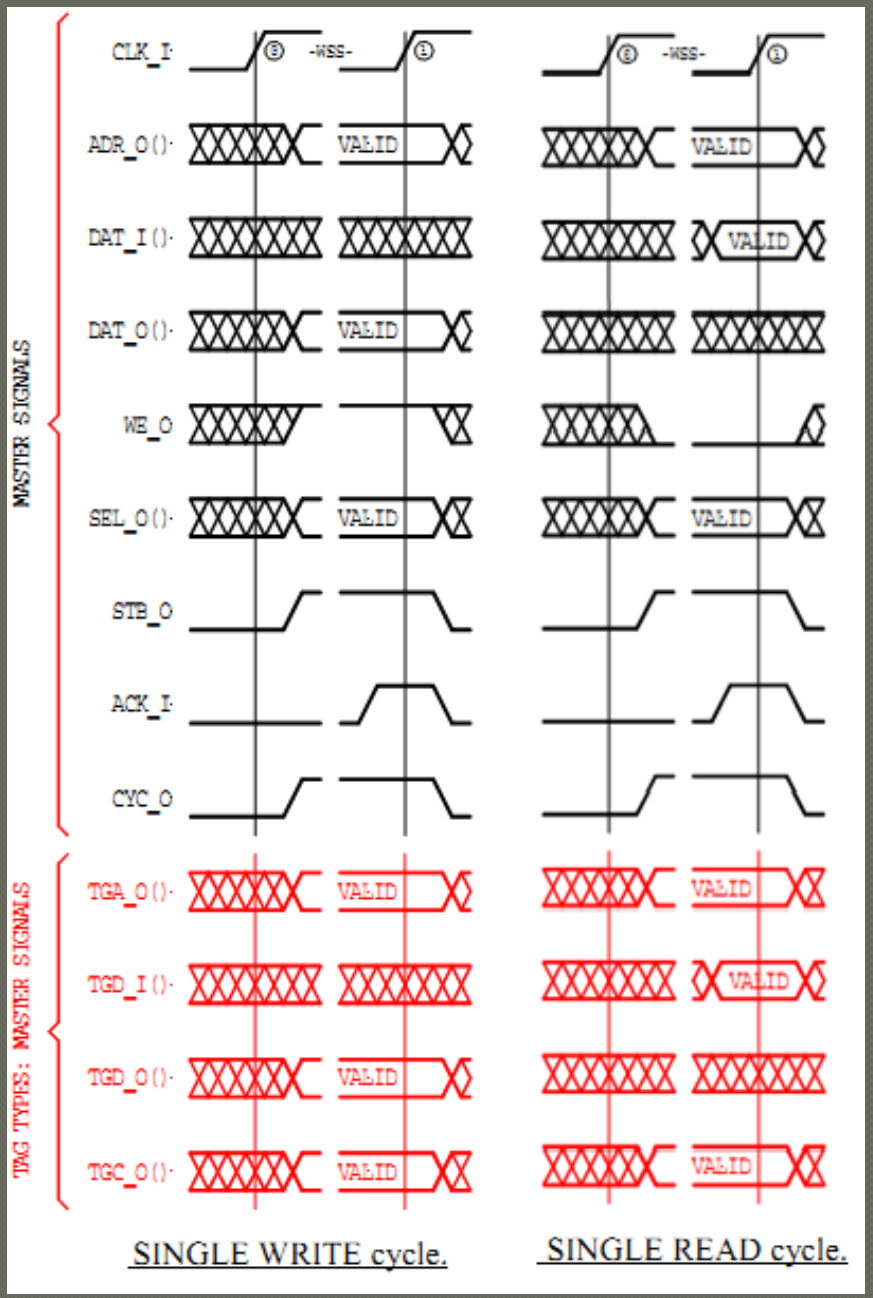



```
procedure EppReadData(
                signal EppWait : in     std_logic;

                signal EppDstb : out    std_logic;
                signal EppWr    : out   std_logic)  is

                constant T1 : Time :=  41 ns;
                -- WAIT low to ASTB or DSTB active min time
                constant T2 : Time :=  81 ns;
                -- ASTB or DSTB active min time

begin

                EppDstb       <= '0';

                report "Waiting for rising edge of EppWait in
                                EppReadData Procedure";
                wait until      EppWait ='1';

                wait for T2;

                EppDstb       <= '1';
                EppWr         <= '1';

                wait for T1;
                 -- WAIT low to ASTB or DSTB active 40ns mini

end EppReadData;
```

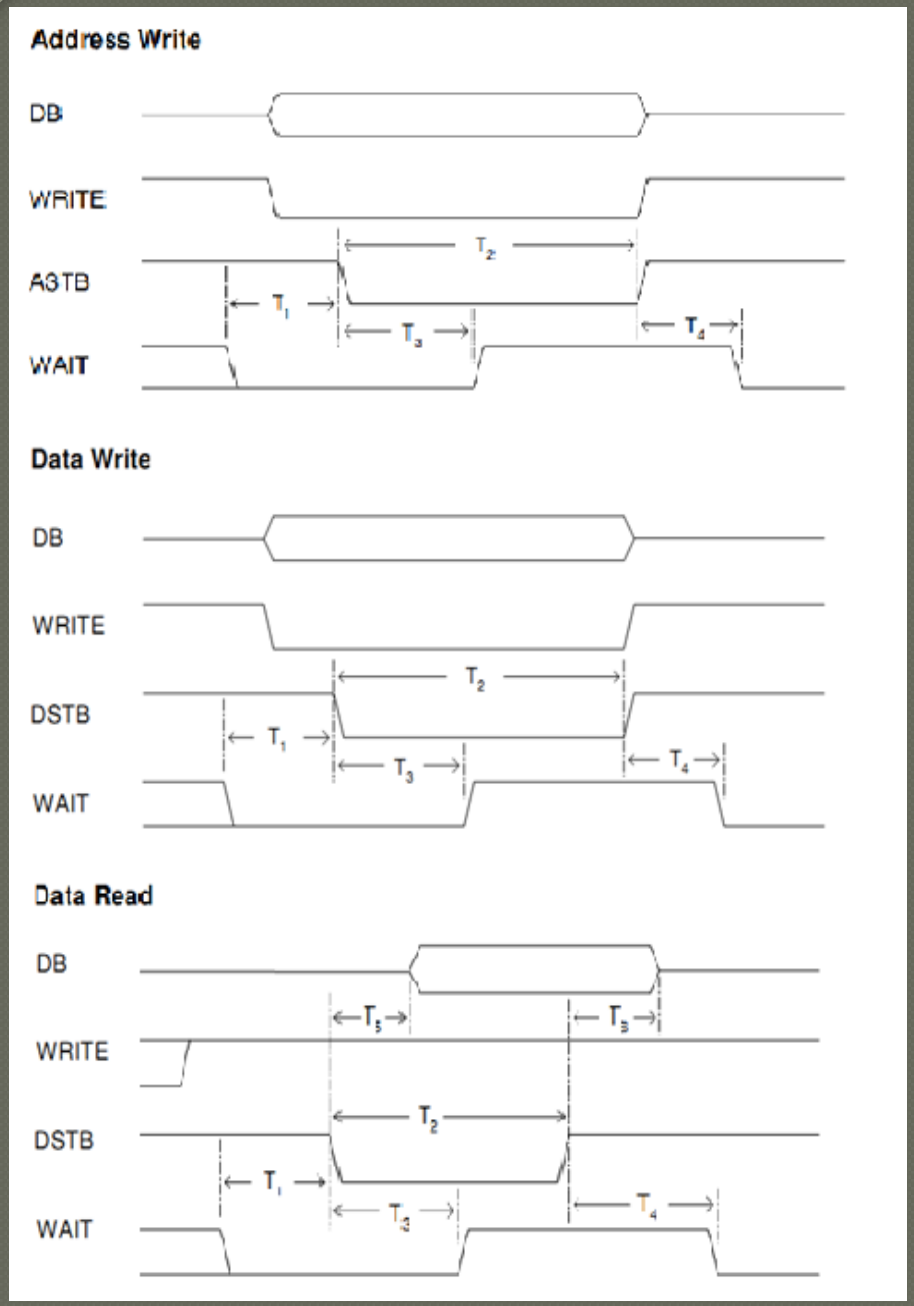
Veamos que tal se ve el diseño en el ISE

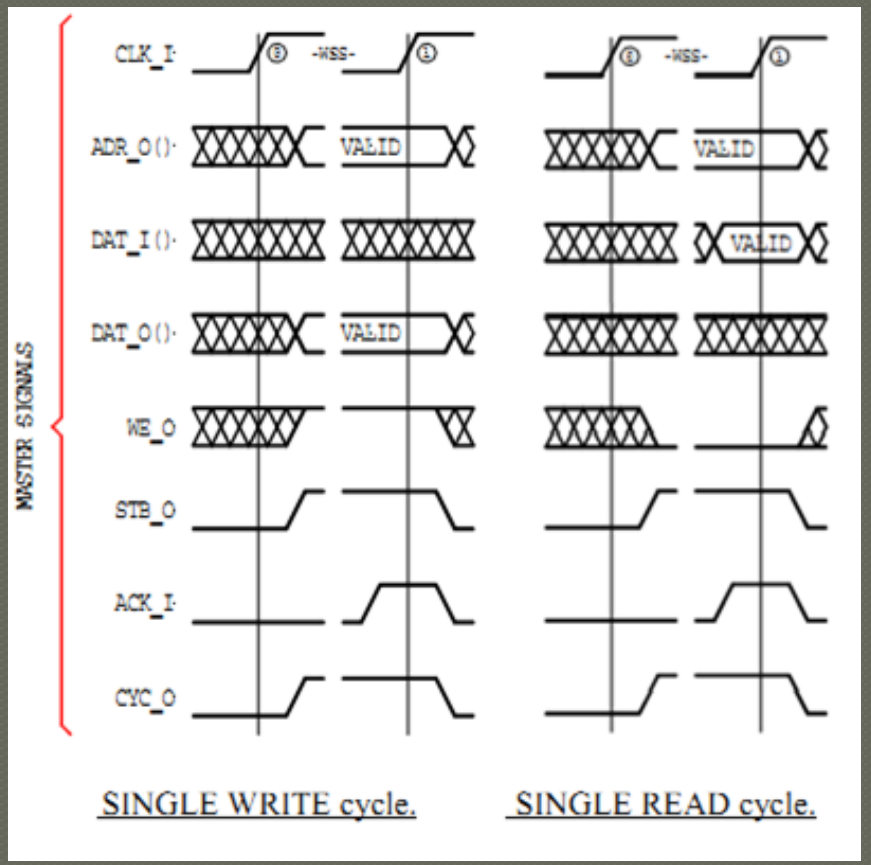
Raúl Arteche Díaz arteche@ceaden.edu.cu Noviembre 2012



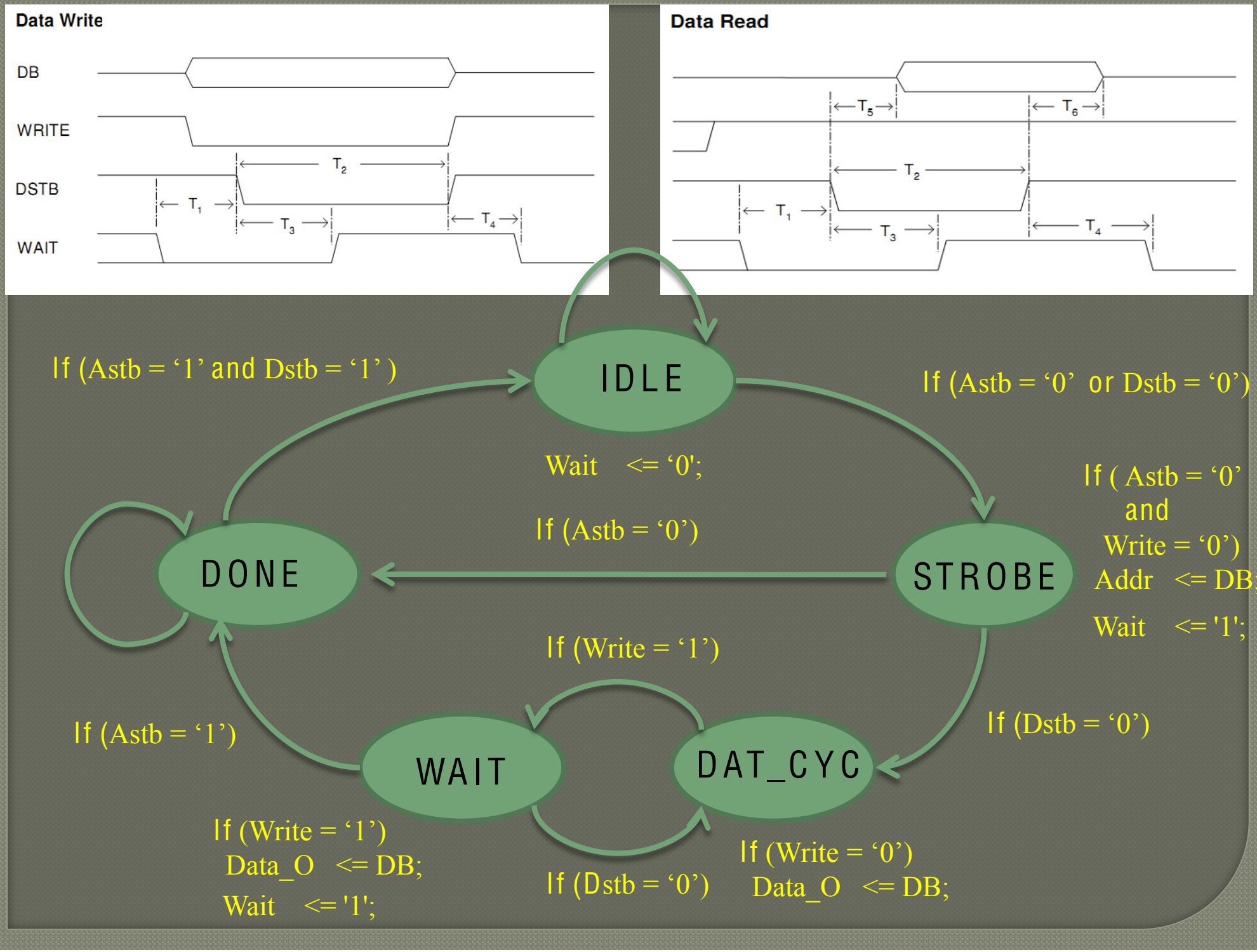


Raúl Arteche Díaz arteche@ceaden.edu.cu Noviembre 2012

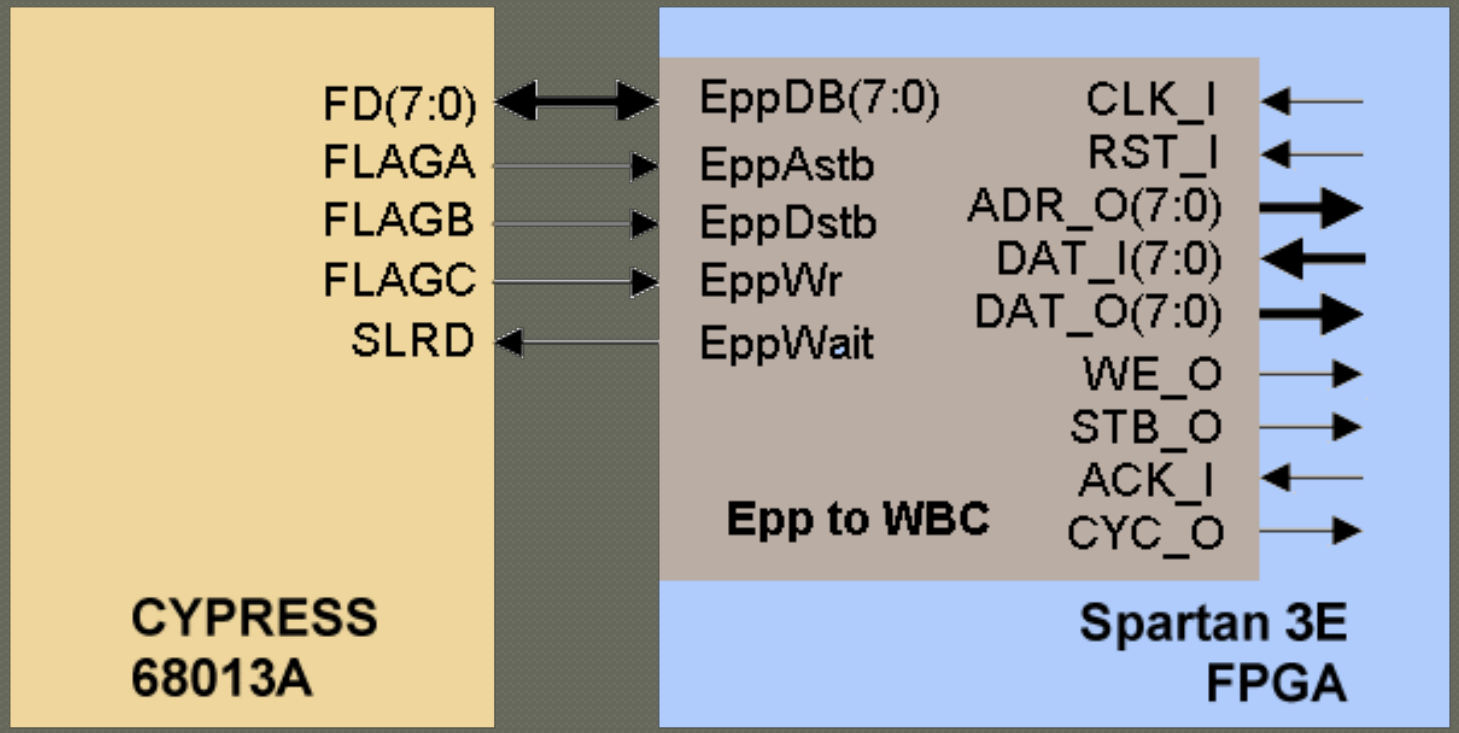




Raúl Arteche Díaz arteche@ceaden.edu.cu Noviembre 2012

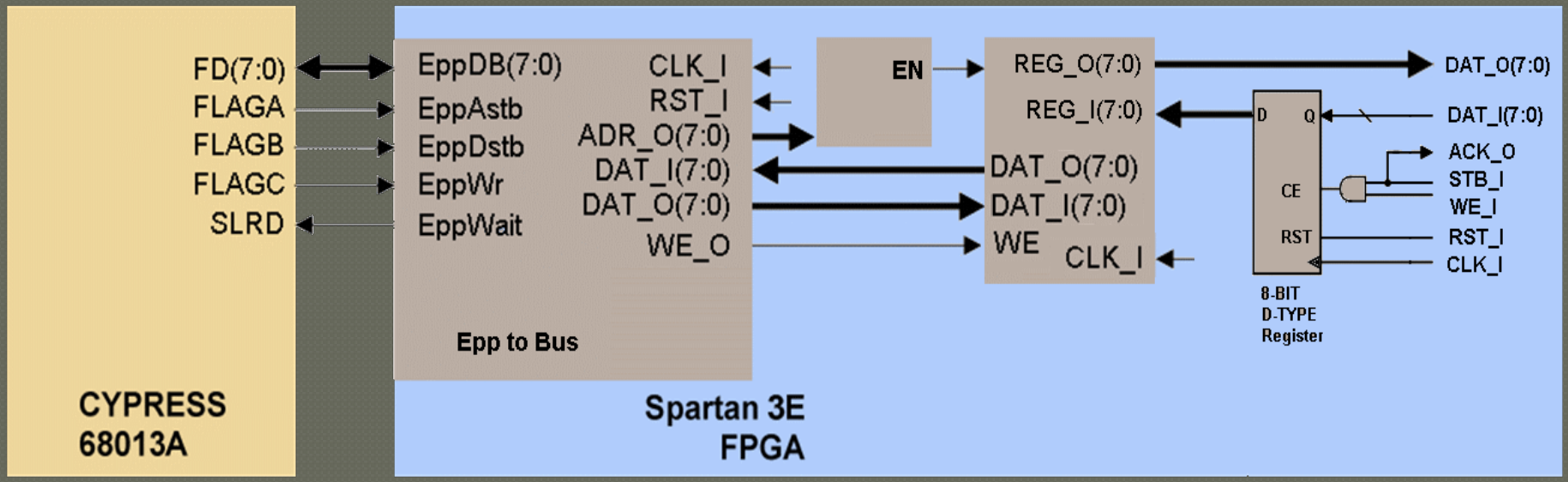

Data Write
DB
WRITE
DSTB
WAIT
T1
T2
T3
T4
Data Read
T5
T6
T1
T2
T3
T4
IDLE
If (Astb = '1' and Dstb = '1')
Wait <= '0';
If (Astb = '0' or Dstb = '0')
If (Astb = '0' and Write = '0') Addr <= DB;
Wait <= '1';
DONE
If (Astb = '0')
STROBE
If (Write = '1')
If (Astb = '1')
If (Dstb = '0')
WAIT
DAT_CYC
If (Write = '1') Data_O <= DB;
Wait <= '1';
If (Dstb = '0')
If (Write = '0') Data_O <= DB;

DEPP a WB Master Interfaz



Raúl Arteche Díaz arteche@ceaden.edu.cu Noviembre 2012

DEPP a WB Slave Interfaz



Raúl Arteche Díaz arteche@ceaden.edu.cu Noviembre 2012