2.4. Descripción de Memorias en VHDL

Memorias en FPGAs

OBJETIVOS:

- Memorias en FPGA: Distribuida o Bloques
- Tipos de escritura en RAM:
  - Write-First
  - Read-First
  - No change
- Descripción VHDL de memoria RAM
- RAM síncrona versus asíncrona
- Descripción VHDL de memoria ROM

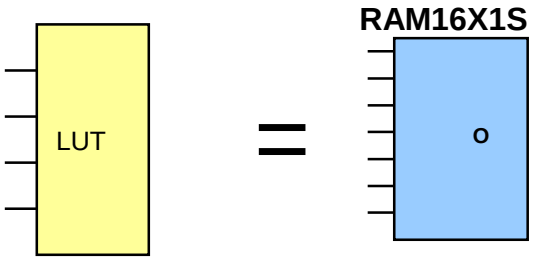
2.4.1. Tipos de memorias en FPGAs

Memorias en FPGAs

- Tipos de memoria en las FPGA
  - RAM distribuida (LUTs)
  - Block RAM (Embebida)
- Instanciación versus Inferencia
  - La **inferencia** permite la **portabilidad**
  - La instanciación optimiza rendimiento y recursos
- Características de las memorias FPGAs
  - Escritura síncrona (en ambos tipos)
  - Lectura:
    - Síncrona (Block RAM)
    - Asíncrona (Distributed RAM)

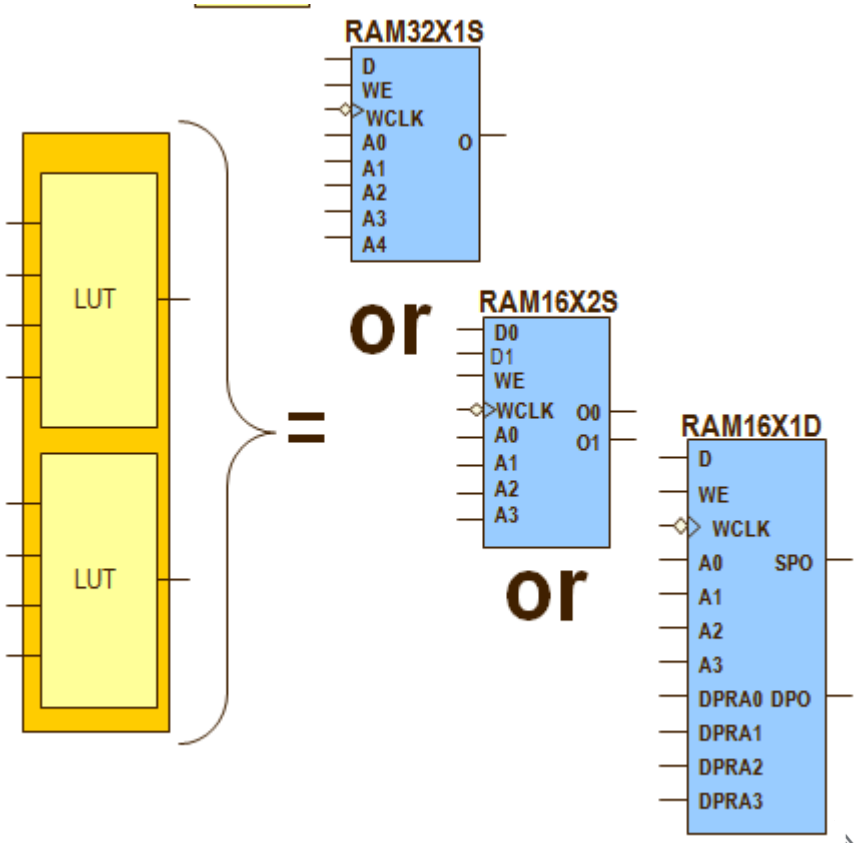
Memorias en FPGAs: RAM Distribuida

- LUT → RAM distribuida
  - Una LUT → 16x1 RAM
  - LUTs en cascada → Amplía el tamaño de la RAM



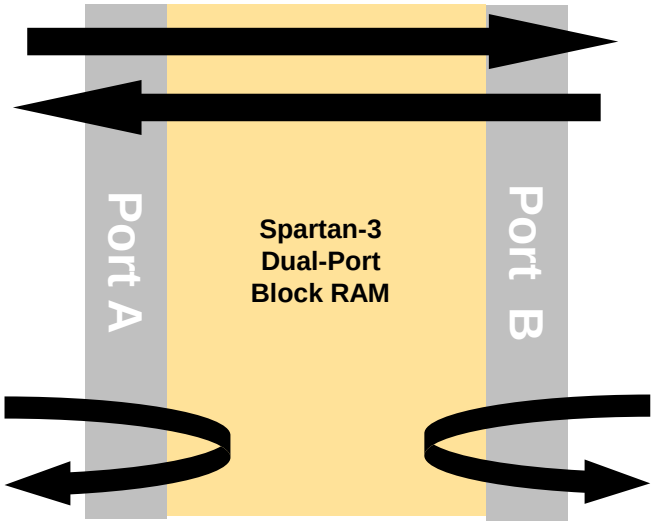
- Lectura asíncrona
  - Síncrona añadiendoles FF-D.

- Dos LUTs pueden formar:
  - 32 x 1 single-port RAM
  - 16 x 2 single-port RAM
  - 16 x 1 dual-port RAM

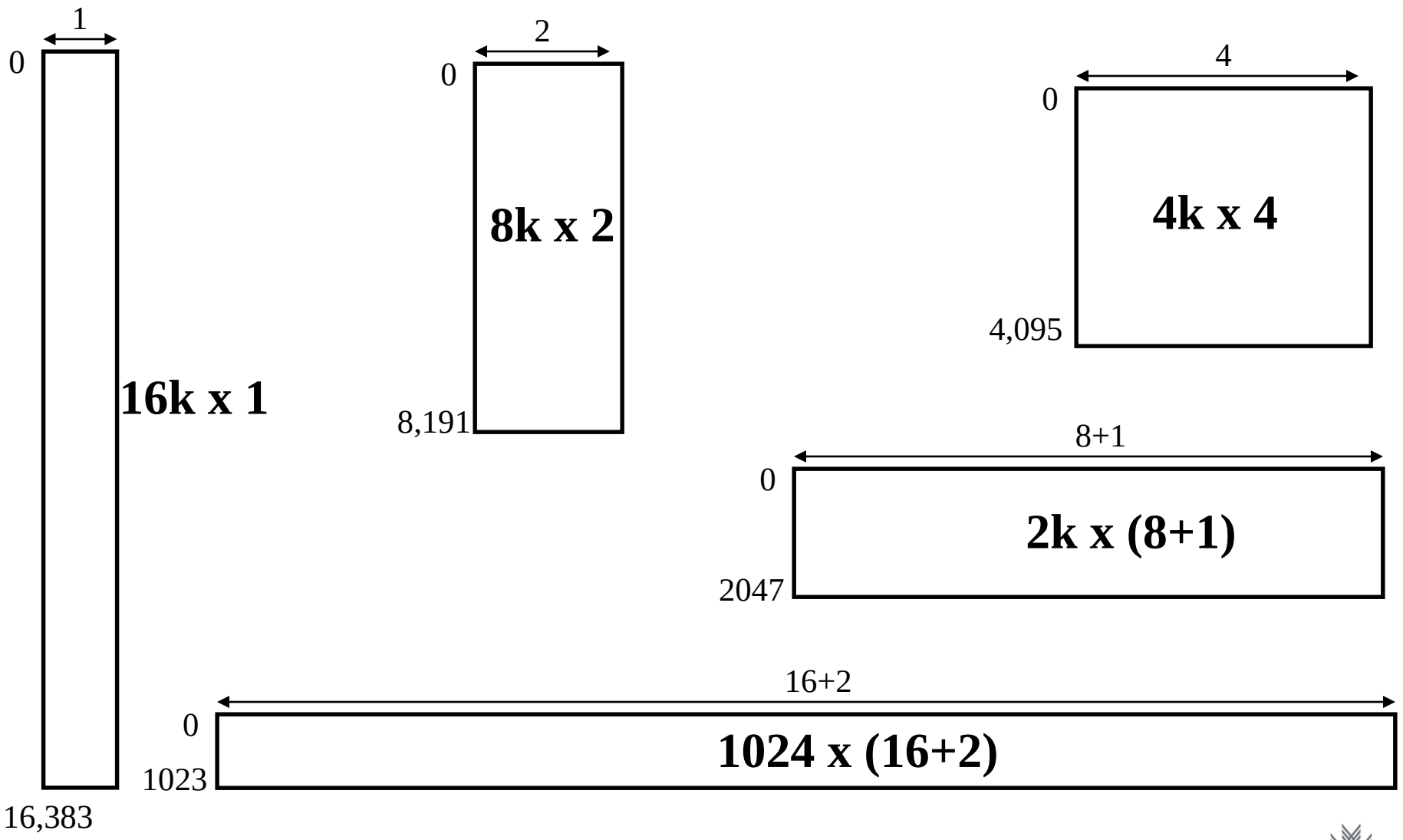


Memorias en FPGAs: Bloques de RAM

- Bloques de memoria RAM dedicado
- Desde 4 hasta más de 100 bloques según modelo
- 18 kbits por bloque (con bit de paridad)
- Se pueden encadenar bloques
- Configurables como de simple o doble puerto
- La lectura y la escritura son siempre **síncrona**



Memorias en FPGAs: Bloques de RAM



2.4.2. Tipo de memoria inferida con HDL

Tipo de memoria RAM inferida

- Block RAM
- Distributed or LUT RAM

Action	Distributed RAM	Block RAM
Write	Synchronous	Synchronous
Read	Asynchronous	Synchronous

Fuente: Xilinx Inc.

Modos de escritura en memoria RAM

Write Mode: Se refiere a la política utilizada cuando se realiza una escritura simultánea con una lectura en una misma dirección (mismo puerto).

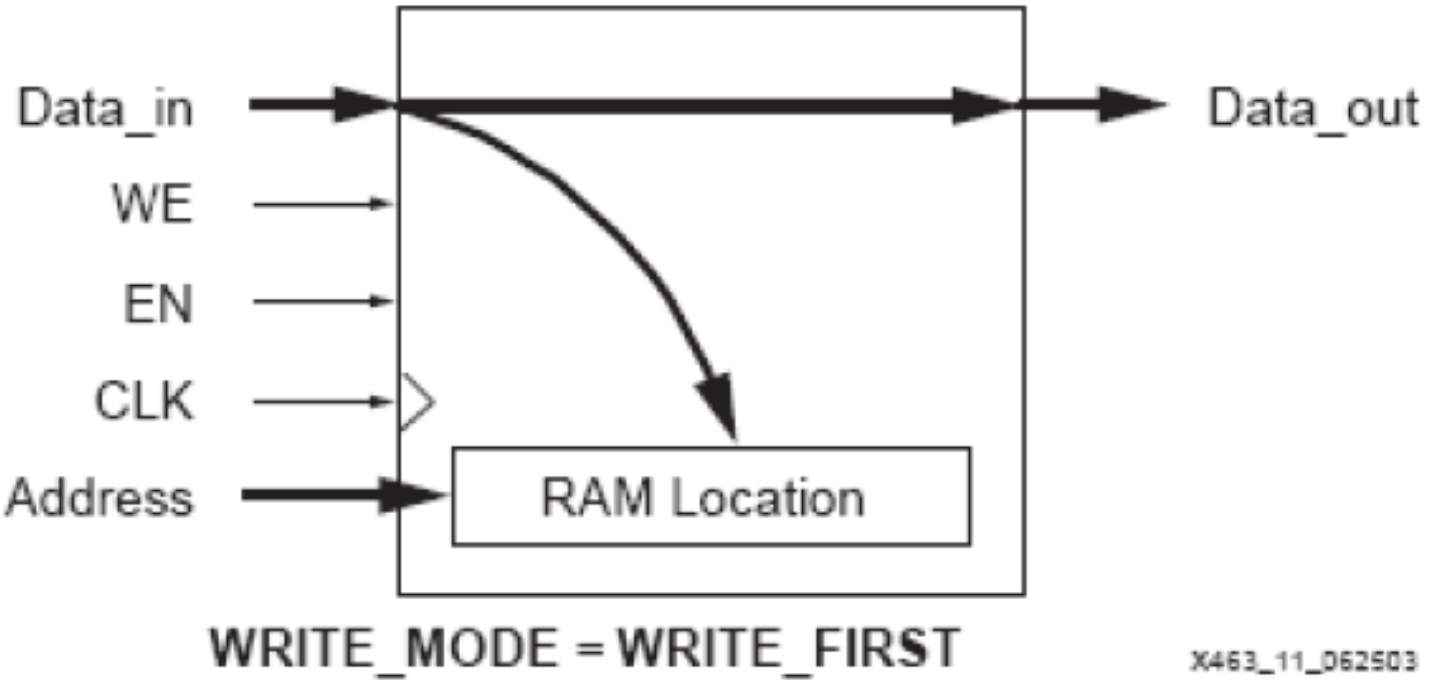
Tipos:

A. **Write First**: El dato de entrada es escrito en la dirección de memoria especificada y simultáneamente está disponible en la salida.

B. **Read First:** Dada la dirección de memoria, el dato guardado en esa dirección aparece en la salida. El dato de entrada, se almacena después en esa dirección.

C. No change: En un ciclo de reloj, o bien se lee, o bien se escribe. Si se escribe, la salida conserva el valor de la anterior operación de solo lectura

A. Modos de escritura WRITE FIRST



Descripción de RAM_WF en VHDL (Block RAM)

A. Modos de escritura WRITE FIRST (Block RAM)

```
entity RAM_WF_MxN is
 generic (      DATA_WIDTH: positive:= 4;
                ADDR_WIDTH: positive:= 10);
 Port ( CLK_i: in STD_LOGIC;
        ADDR_RAM_i : in STD_LOGIC_VECTOR (ADDR_WIDTH-1 downto 0);
        DATA_RAM_i : in STD_LOGIC_VECTOR (DATA_WIDTH-1 downto 0);
        DATA_RAM_o : out STD_LOGIC_VECTOR(DATA_WIDTH-1 downto 0);
        WE_i : in STD_LOGIC);
end RAM_WF_MxN;
```

PROJECT_23

No añadir RESET asíncrono

A. Modos de escritura WRITE FIRST (Block RAM)

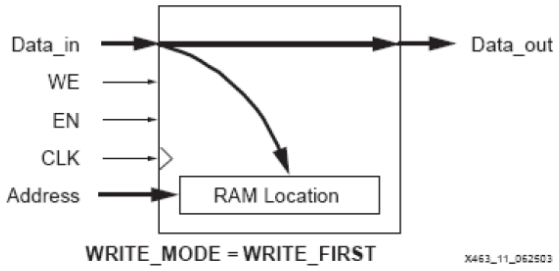
```
architecture behavioral of RAM_WF_MxN is
        type RAM_TYPE is array(2**ADDR_WIDTH-1 downto 0) of
                        std_logic_vector(DATA_WIDTH-1 downto 0);
        signal RAM : RAM_TYPE;
begin
process (CLK_i)
    begin
     if rising_edge(CLK_i) then
         -- Operación de LECTURA/ESCRITURA síncrona
        if WE_i = '1' then
           RAM(to_integer(unsigned(ADDR_RAM_i))) <= DATA_RAM_i ;
        -- El dato que se lee, es el mismo que se escribe
         DATA_RAM_o <= DATA_RAM_i;

        else
        -- Operación de SOLO LECTURA SÍNCRONA
         DATA_RAM_o <= RAM(to_integer(unsigned(ADDR_RAM_i)));
        end if;

      end if;
end process;
end Behavioral;
```

1

2

3

4



Tipos de datos en VHDL: Datos definidos por el usuario (type)

1

En VHDL el usuario puede definir tipos de datos nuevos:

```
type <nombre_tipo> is <tipo_de_dato>;
type Cero_Siete is unsigned(2 downto 0);--0,1,2,…,7
```

2

En VHDL el usuario puede definir un array de elementos de un tipo compuesto:

```
type <nombre_array> is array <rango_array> of <tipo_vector>;
type ram_type is array(15 downto 0) of
                              std_logic_vector(3 downto 0);
```

ram_type

15	“0110”
14	“1110”
…	
1	“0000”
0	“0101”

Conversión entre tipos de datos en VHDL: std_logic to integer

3

En VHDL el tipo de datos std_logic_vector puede convertirse a integer, **previa conversión** de std_logic_vector a signed/unsigned.

Aplicación: actuar como índice en arrays de elementos.

```
RAM(to_integer(unsigned(Address))) <= DataIn ;
```

A. Modos de escritura WRITE FIRST (Block RAM)

Tras la síntesis…

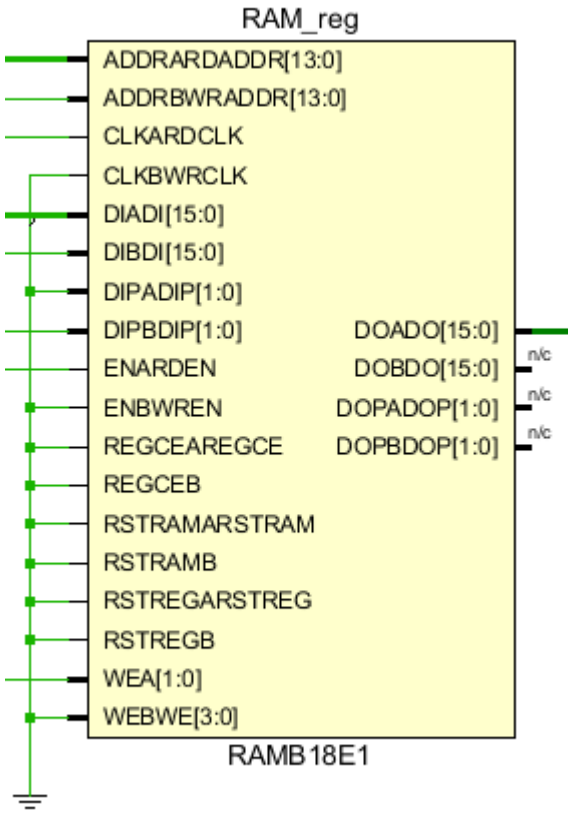
Tcl Console | Messages | Log | Reports | Design Runs | **Utilization** | Package Pin

- Hierarchy
- Summary
- Slice Logic
- Memory
  - Block RAM Tile (1%)
    - RAMB18 (1%)
      - RAMB18E1 only
- DSP
- IO and GT Specific
  - Bonded IOB (10%)
- Clocking
  - BUFGCTRL (3%)
- Specific Feature
- Primitives

Primitives

Ref Name	Used	Functional Category
IBUF	16	IO
OBUF	4	IO
RAMB18E1	1	Block Memory
BUFG	1	Clock

Solo usa BRAM si se precisa una memoria grande, si no, usa LUT-RAM



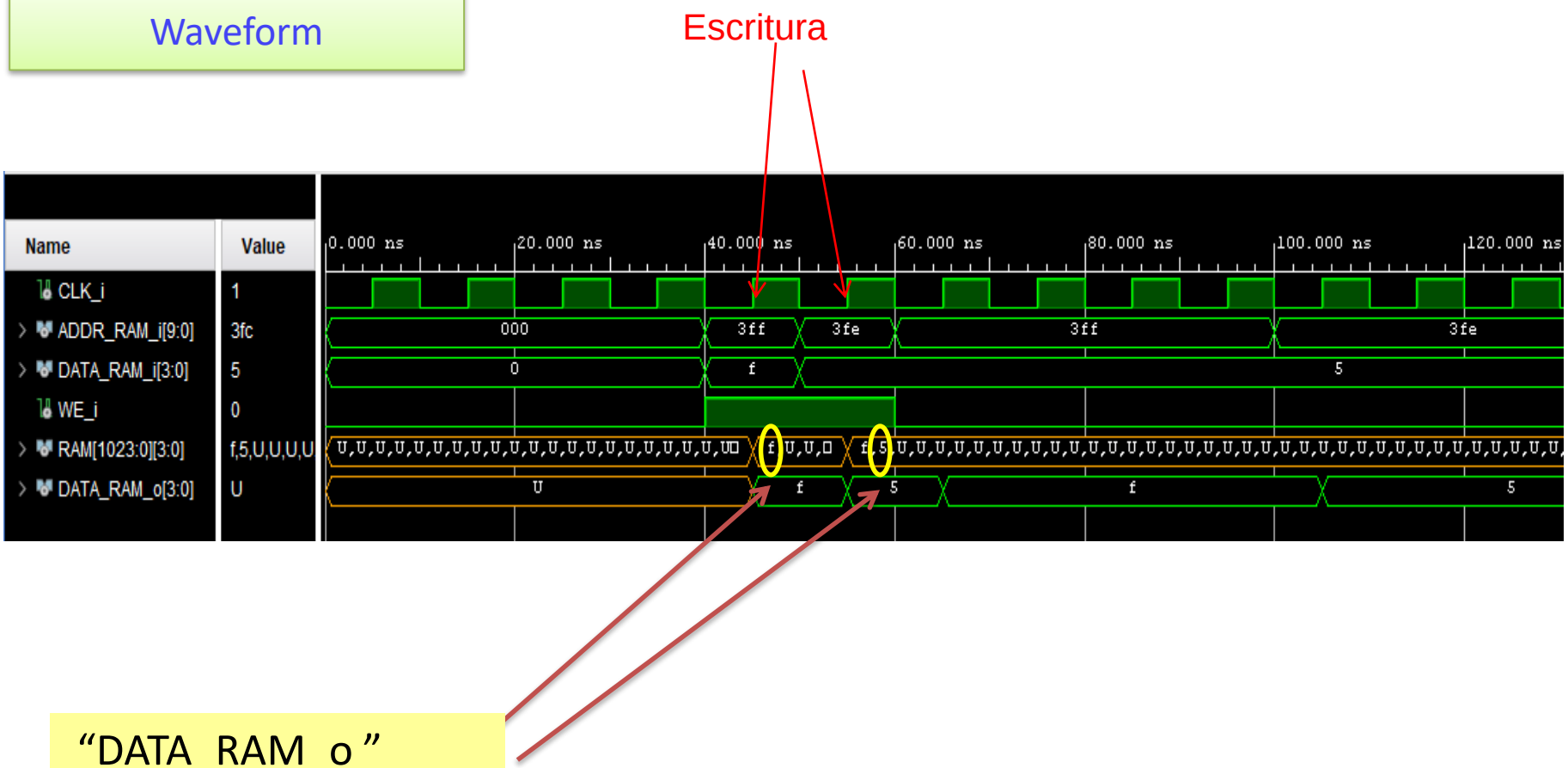
A. Modos de escritura WRITE FIRST (Block RAM)

Testbench

```
stimulus: process
 begin
  ADDR_RAM_i <= (others => '0');
  DATA_RAM_i <= (others => '0');
  WE_i <= '0';
  wait for 40 ns;
    -- WRITING ------
    ADDR_RAM_i <= "1111111111";
    DATA_RAM_i <= (others => '1');
    WE_i <= '1';
    wait for 10 ns;

    ADDR_RAM_i <= "1111111110";
    DATA_RAM_i <= "0101";
    WE_i <= '1';
    wait for 10 ns;

    -- READING --------
    WE_i <= '0';
    ADDR_RAM_i <= "1111111111";
    wait for 40 ns;
    ADDR_RAM_i <= "1111111110";
    wait for 40 ns;
    ADDR_RAM_i <= "1111111100";
    wait for 40 ns;

  stop_the_clock <= true;
  wait;
 end process;
```

A. Modos de escritura WRITE FIRST (Block RAM)

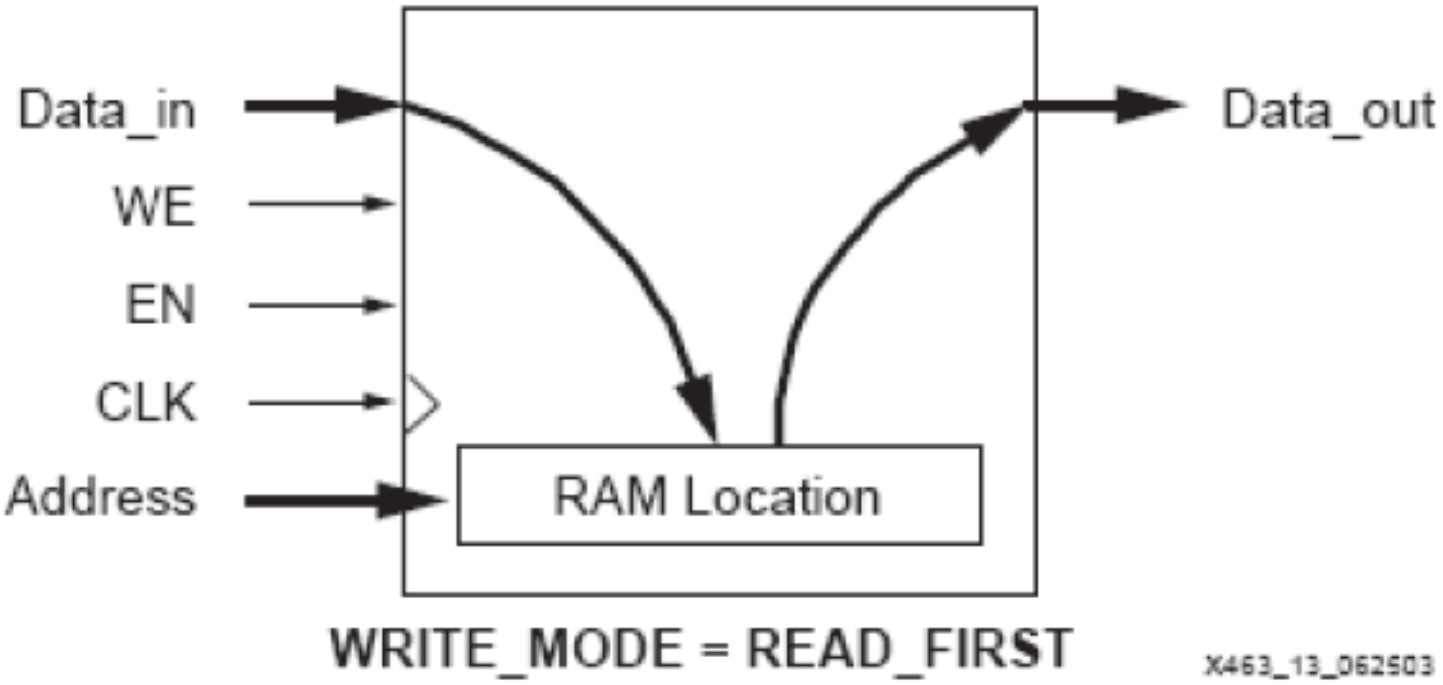
Waveform



"DATA_RAM_o" toma el mismo valor que se está escribiendo en el mismo flanco de CLK

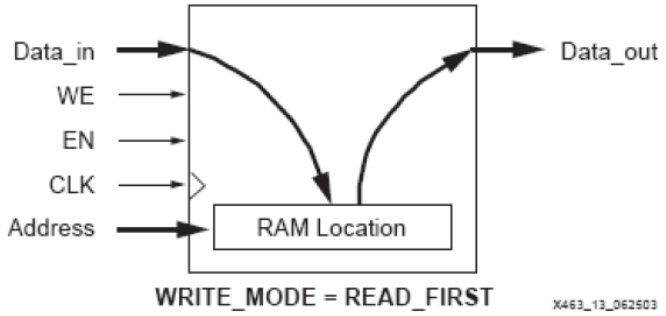
Descripción de RAM_RF en VHDL (Block RAM)

B. Modos de escritura READ FIRST (Block RAM)

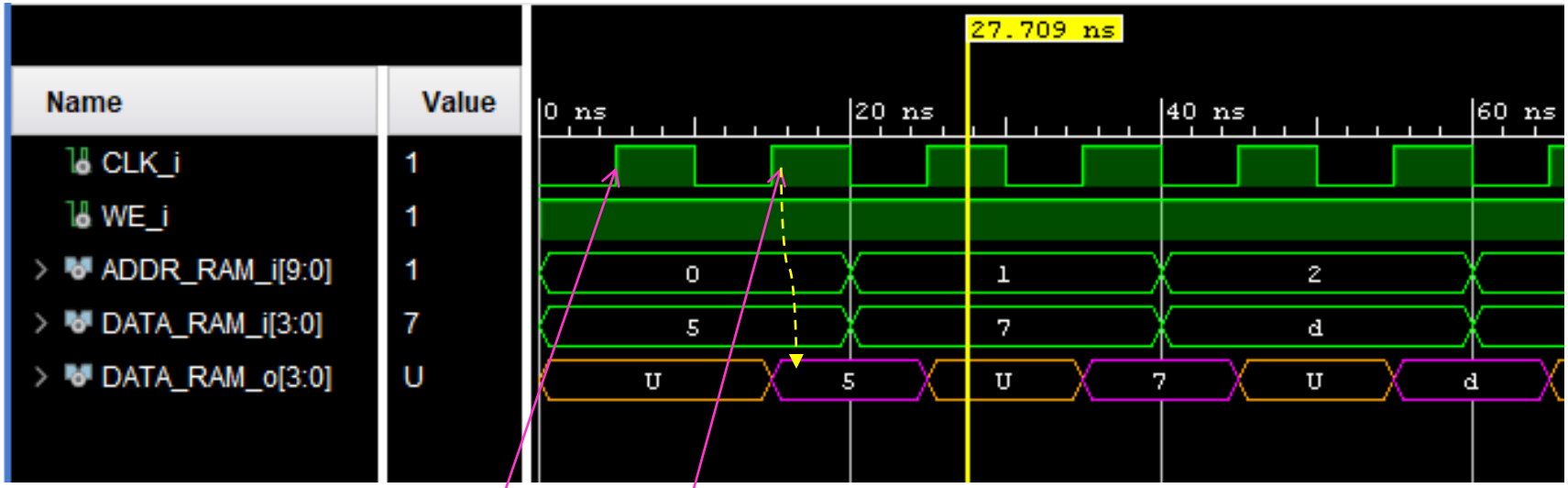


B. Modos de escritura READ FIRST (Block RAM)

```
process (CLK_i)
Begin
 if rising_edge(CLK_i) then
        -- ESCRITURA síncrona
       if WE_i = '1' then
        RAM(to_integer(unsigned(ADDR_RAM_i))) <= DATA_RAM_i;
       end if;

       -- LECTURA SÍNCRONA
       DATA_RAM_o <= RAM(to_integer(unsigned(ADDR_RAM_i)));

 end if;
end process;
```



B. Modos de escritura READ FIRST (Block RAM)



Escritura

Lectura

"DAATA_RAM_o" NO toma el mismo valor que se está escribiendo hasta el siguiente ciclo. Mientras, muestra el valor anteriormente almacenado, 'U' en este caso.

Descripción de RAM en VHDL (Distributed RAM)

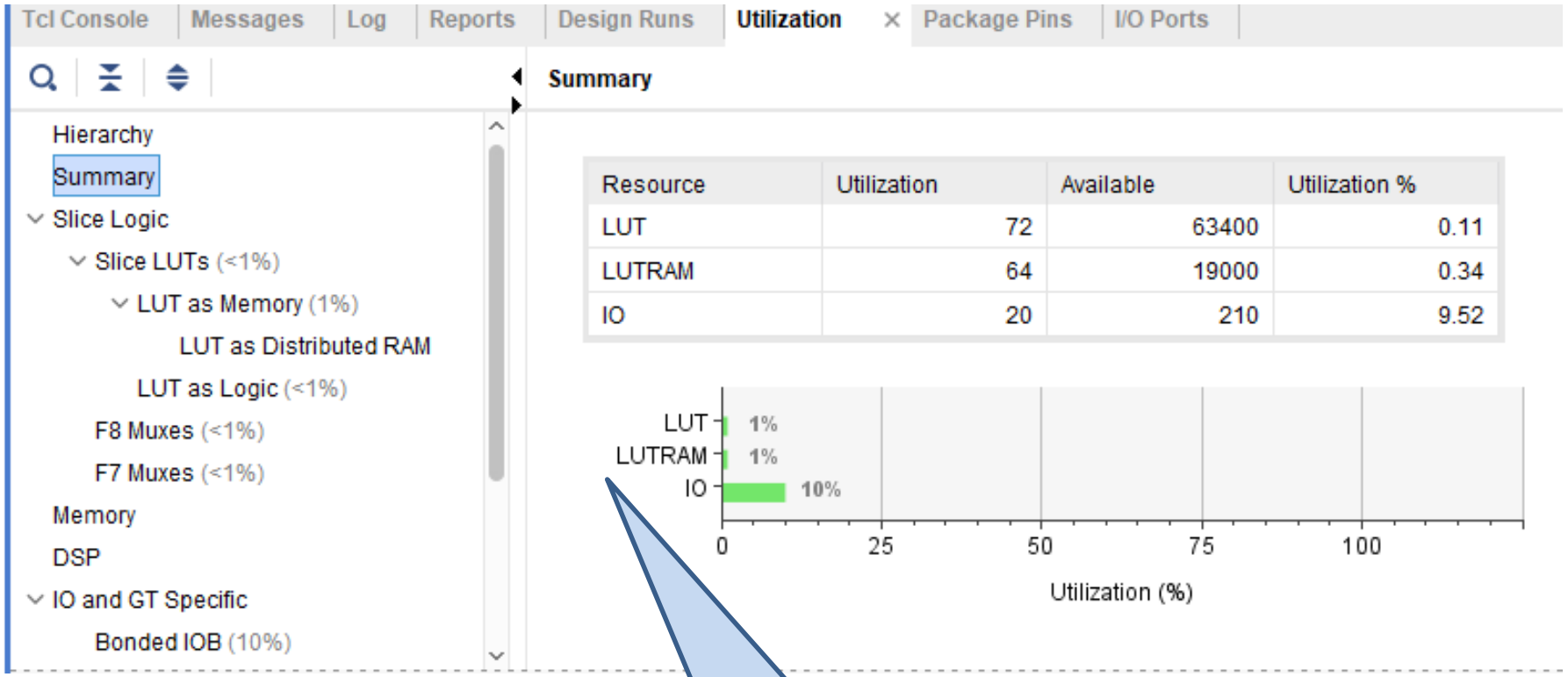
Descripción de RAM con lectura asíncrona (Distributed RAM)

PROJECT_23

```
architecture Behavioral of RAM_WF_ASYNC is
-- USers type declaration
type RAM_TYPE is array((2**ADDR_WIDTH)-1 downto 0) of
std_logic_vector(DATA_WIDTH-1 downto 0);
signal RAM : RAM_TYPE;

begin
-- Distributed RAM
RAM_MEM:process(CLK_i)
 begin
  if rising_edge(CLK_i) then
        if WE_i = '1' then
         -- Write operation (Distributed, Sync Write)
         RAM(to_integer(unsigned(ADDR_RAM_i))) <= DATA_RAM_i ;
        end if;
  end If;
end process;

-- Read operation (Distributed, Asyncronous Read)
        DATA_RAM_o <= RAM(to_integer(unsigned(ADDR_RAM_i)));
end Behavioral;
```

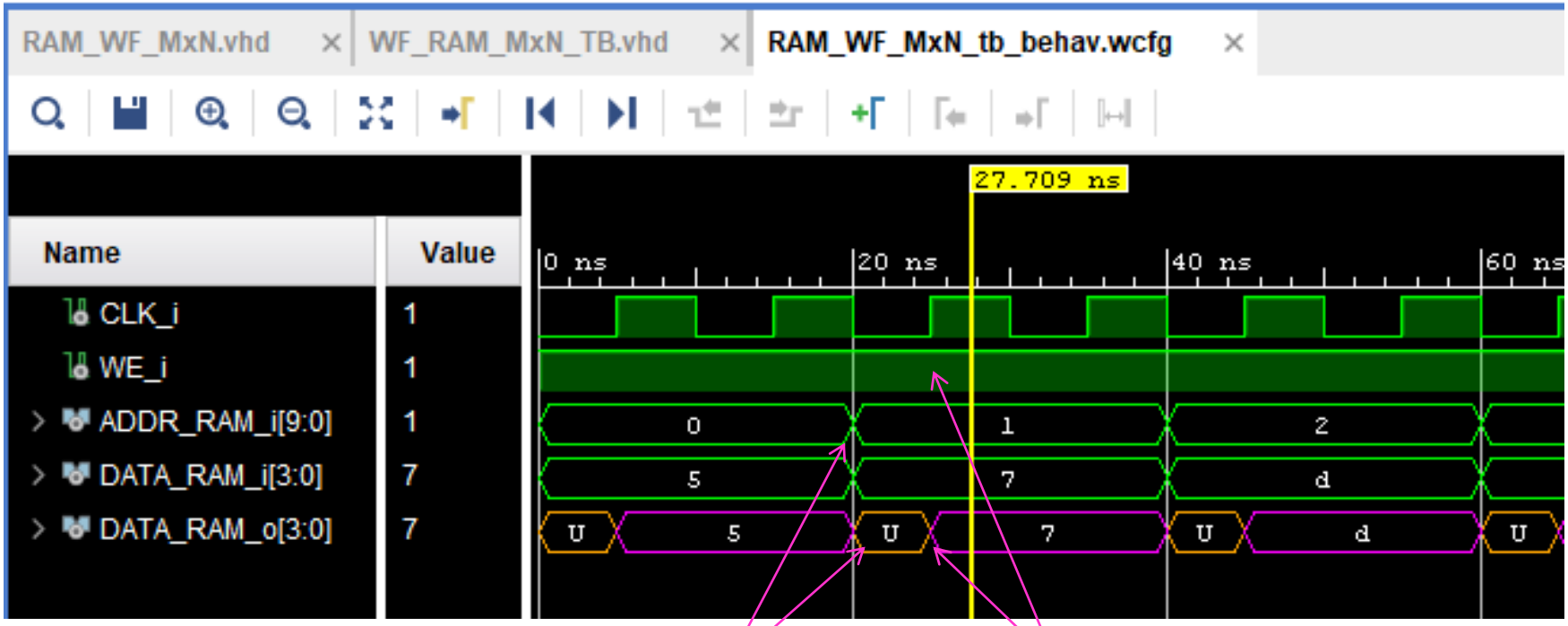
External to the process

Descripción de RAM con lectura asíncrona (Distributed RAM)



RAM distribuida → Implementada con LUT

Descripción de RAM con lectura asíncrona (Distributed RAM)



READ: Cambia "DATA_RAM_o" porque hay un cambio de dirección "ADDR_RAM_i". Independientemente de que haya un flanco positivo de reloj (Asíncrona)

WRITE: Escribe el dato en la dirección indicada cuando llega el flanco positivo de reloj. "DATA_RAM_o" también se actualiza con ese nuevo dato

2.4.3. Descripción de ROM en VHDL

Descripción VHDL de Memoria ROM

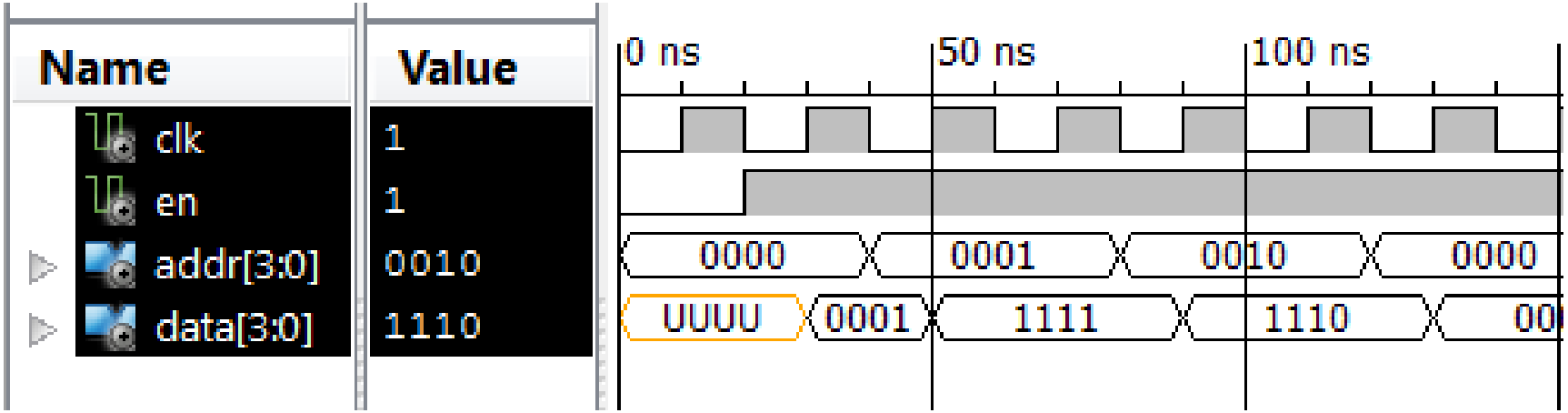
```
entity ROM_16x4 is
 port (   CLK_i: in std_logic;
          ENA_i:  in std_logic;
          ADDR_BUS_i: in std_logic_vector(3 downto 0);
          DATA_BUS_o:out std_logic_vector(3 downto 0));
end ROM_MxN;

architecture behavioral of ROM_MxN is
     type ROM_TYPE is array (15 downto 0) of std_logic_vector(3 downto 0);
     signal ROM : ROM_TYPE :=(X"1",X"2",X"3",X"4",X"5",
     X"6",X"7",X"8",X"9",X"A",X"B",X"C",X"D",X"E",X"F",X"1");

begin

process(CLK_i)
begin
     if rising_edge(CLK_i) then
          if ENA_i = '1' then
           DATA_BUS_o <= ROM(to_integer(unsigned(ADDR_BUS_i)));
          end if;
     end if;
end process;

end behavioral;
```

1

X"A" → Formato Hex

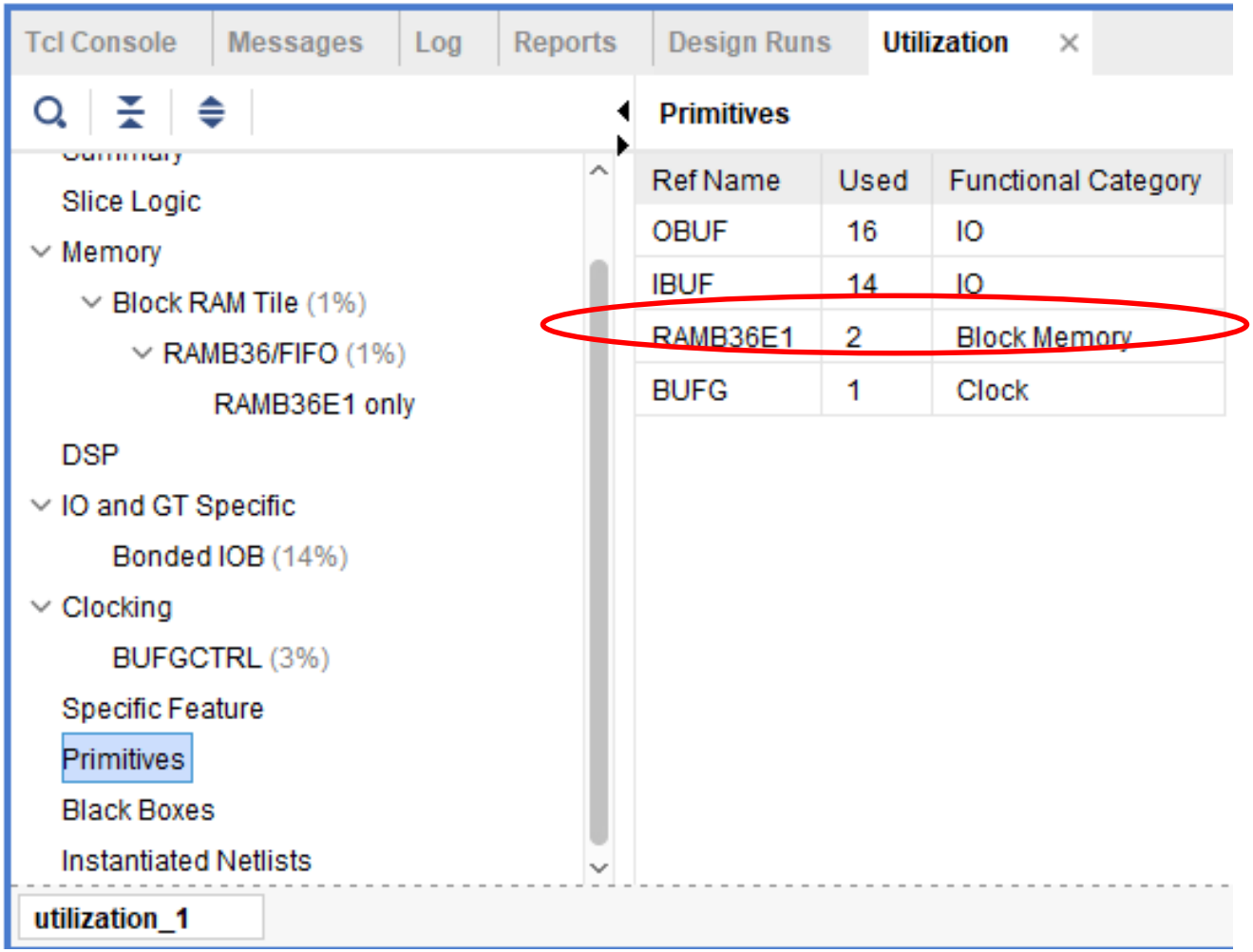
Enable='0' conserva el valor anterior

Descripción VHDL de Memoria ROM



Descripción VHDL de Memoria ROM

Synthesis Report



Bibliografía

- **VHDL FOR LOGIC SYNTHESIS.** Andrew Rushton. 2011 John Wiley & Sons, Ltd. Published
- **Free range VHDL**. Bryan Mealy, Fabrizio Tappero. (Creative Commons). http://www.freerangefactory.org (Mayo 2013)
- **VHDL 101.** William Kafig. Editorial Elsevier. 2011