UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA

ESCUELA PROFESIONAL DE INGENIERÍA ELECTRÓNICA

LABORATORIO N°7

Curso: Laboratorio de sistemas digitales II

Docente: Ing. Palomino Peña Celso

Semestre: 2024-I

Alumnos:

Lujan Edilia Huamanga Chumbes	201250
Jesus Leonardo Ima Chuquichampi	200834
Bremdow Salazar Roa	200353
Aaron Coyla Quispe	200832

2024

Introducción

En un inicio el almacenamiento de información se realizaba de forma analógica mediante el uso de papel como el método mas confiable para poder registrar información con el paso del tiempo y el desarrollo tecnológico el almacenamiento paso a ser digital mediante el empleo de materiales semiconductores capaces de almacenar un bit como la mínima y más básica unidad de almacenamiento.

Con el nacimiento de este innovador método de almacenamiento con el tiempo se fueron desarrollando dispositivos capaces de almacenar grandes cantidades de información en el rango de los gigabytes, terabytes, etc. Conforme se fueron agregando más dispositivos de almacenamiento.

En la actualidad existen diferentes tipos de memorias según el propósito deseado dentro de estas se encuentra las RAM, ROM, PROM, EPROM las cuales tienen diferentes características siendo la ROM una memoria estática o de solo lectura.

Marco Teórico

Memorias ROM

Las memorias ROM son dispositivos del almacenamiento de solo lectura esto implica que la información una vez grabada esta no puede modificarse o se requiere de procesos específicos según el tipo de memoria ROM.

La forma en la que una memoria ROM almacena información es mediante una matriz de dispositivos básicos de información siendo así que estas organizan con una longitud de palabra de 8 bits o 1 byte y una cantidad de direcciones equivalente a una potencia de 2 que fueron el estándar creado para ambas propiedades antes mencionadas.

En la práctica una ROM esta constituida por una matriz de dispositivos unipolares (MOS) o bipolares (BJT) los cuales se encuentra en corte o saturación para representar uno de los estados lógicos (1 o 0), en la figura 1 se muestra la estructura de la ROM de este tipo. (?, ?)

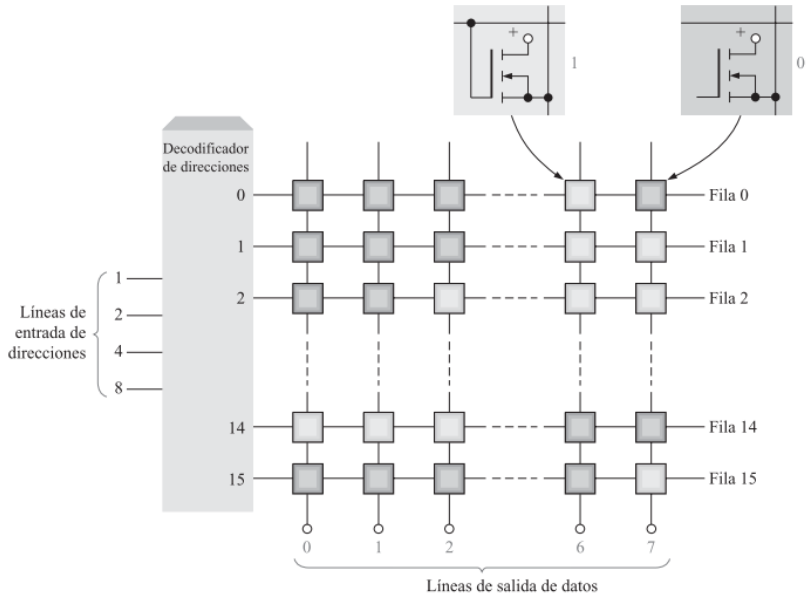

Figura 1

Representación Matricial de una ROM de 16x8 bits

Familias ROM

Una memoria ROM se sub-divide en diferentes categorías según la tecnología y método a usar para la escritura o en caso muy particulares la re-escritura de datos, dentro

de esta clasificación, se tiene:

- ROM de máscara
- PROM (ROM Programable)
- EPROM (ROM Programable Borrable)
- UV PROM (ROM Programable Borrrable mediante Rayos Ultravioleta)
- EEPROM (ROM Programable Borrable Electricamente)

Aplicaciones de la ROM

Las ROM son ampliamente utilizadas para el almacenamiento de información de forma no volátil lo que implica que este tipo de memorias no requieren y/o dependen de una fuente de alimentación constante para mantener la información previamente grabada.

Simulación

Para poder emular el comportamiento de una memoria ROM se hizo uso del software Quartus en su versión V.20.1 la cual permitió modelar este componentes lógicos mediante el lenguaje de descripción de hardware VHDL.

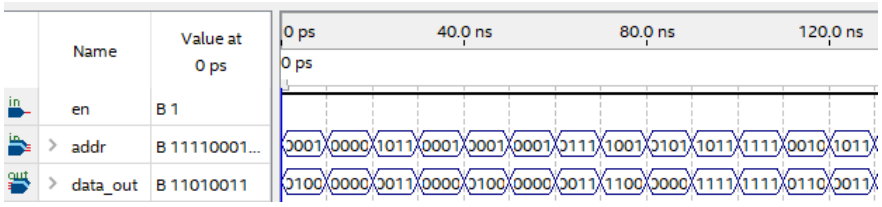

Figura 2

Forma de Onda de salida para la ROM 8x4

Como se puede apreciar en le 2 se puede apreciar que se tiene una salida o lectura de datos en función a una dirección de memoria y su vez esta salida se encuentra en dependencia de la entrada habilitadora que permite el acceso a cada una de las memorias ROM 4x2.

Resultados

A lo largo del desarrollo y evolución tecnológica, las memorias ROM han demostrado ser una solución efectiva para el almacenamiento no volátil de datos, donde la

información una vez grabada no puede modificarse de forma sencilla siendo necesario para este proceso el uso de procesos específicos.

Las diferentes variantes de la memoria ROM, como la PROM, EPROM, y EEPROM, han permitido que estas memorias se adapten a diversas necesidades tecnológicas, desde el almacenamiento de código fijo en dispositivos electrónicos hasta la posibilidad de reprogramación de estas memoriaas para ciertas aplicaciones.

Finalmente, la simulación de memorias ROM realizada mediante el software Quartus II, utilizando el lenguaje de descripción de hardware VHDL, el cual ha permitido modelar y entender mejor el comportamiento de los módulos ROM, así como su expansión a partir de encapsulados de menor capacidad de palabra o de direcciones.

Recomendaciones

- La definición VHDL de memorias ROM de gran cantidad de direcciones o tamaño requiere bastante recursos computaciones por lo que es preferible hacer uso de herramientas ya definidas como lo son las librerías de memoria, las cuales optimizan los recursos de hardware.
- Para un análisis más complejo y profundo del comportamiento de una ROM se puede tratar este tema desde la perspectiva visual en función a bloques lo cual brinda una mayor comprensión del funcionamiento al realizar las expansiones y definiciones de cada memoria de forma gráfica.

Bibliografía

1] Floyd, T. L. (2006). Fundamentos de sistemas digitales (9.ª ed.). Pearson Educación

Anexos

```vhdl
-- Library
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


-- Entidad
entity rom4x2 is
port(
    en : in  std_logic;
    addr : in std_logic_vector(1 downto 0);
    data_out: out std_logic_vector(1 downto 0)
);
end entity;

-- Arquitectura
architecture arch of rom4x2 is
-- Definiendo la memoria
type data_array is array (0 to 3) of std_logic_vector(1 downto 0);
constant rom : data_array := (
            "01", "10", "11",  "11"
        );
begin
    process(en)
    begin
        if en = '1' then
            data_out <= rom(to_integer(unsigned(addr)));
        else
            data_out <= "00";
        end if;
    end process;
end architecture;
```

Figura 3

Módulo ROM 4x2

```vhdl
-- Library
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


-- Entidad
entity rom8x4 is
port(
    en : in  std_logic;
    addr : in std_logic_vector(11 downto 0);
    data_out: out std_logic_vector(7 downto 0)
);
end entity;

-- Arquitectura
architecture arch of rom8x4 is
-- Importando el componente
component rom4x2 is
port(
    en : in  std_logic;
    addr : in std_logic_vector(1 downto 0);
    data_out: out std_logic_vector(1 downto 0)
);
end component;
begin
    R1 : rom4x2 port map (addr(11), addr(10 downto 9), data_out(7 downto 6));
    R2 : rom4x2 port map (addr(8), addr(7 downto 6), data_out(5 downto 4));
    R3 : rom4x2 port map (addr(5), addr(4 downto 3), data_out(3 downto 2));
    R4 : rom4x2 port map (addr(2), addr(1 downto 0), data_out(1 downto 0));
end architecture;

```

Figura 4

Memoria ROM 8x4 definida de forma estructural