UNIVERSIDAD NACIONAL DE TUCUMÁN

FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍA

ELECTRÓNICA II

Actividad 8: Memorias

Autor: Chaves, Gonzalo Efraín - Estudiante de Ing. Electrónica

Resumen

En esta actividad se investigaron las tecnologías de memoria semiconductora, así como sus aplicaciones dentro de la jerarquía de memoria de un sistema de cómputo, incluyendo conjuntos de registros, memoria caché, memoria principal y memoria de almacenamiento.
A partir de esta investigación se desarrollaron descripciones de hardware en VHDL para distintos arreglos de memoria, asi como una practica de laboratorio con la implementacion en una FPGA de la familia iCE40.
Para cada módulo se elaboraron bancos de prueba automáticos que permiten verificar el funcionamiento y la correcta inferencia de bloques de memoria dedicados (BRAM) durante la síntesis. También se analizó el proceso de inicialización desde archivos externos y las consideraciones necesarias para que el sintetizador reconozca las estructuras como memorias reales.
Los resultados obtenidos permitieron comprender tanto el funcionamiento interno de las tecnologías de almacenamiento como los aspectos prácticos de su descripción en HDL para dispositivos programables.

Introducción

Las memorias semiconductoras son dispositivos digitales utilizados para almacenar datos mediante circuitos integrados basados en materiales como el silicio. La unidad mínima de almacenamiento es la celda de memoria, capaz de retener un único bit. Estas celdas se organizan en matrices bidimensionales de filas (palabras) y columnas, accesibles mediante un decodificador de direcciones.

Las celdas de memoria se organizan en una matriz bidimensional de filas y columnas(ver figura 1.1 y 1.2):

1. Palabra (Word): Una fila completa de la matriz. Si una memoria es de 32 bits de ancho, la palabra tiene 32 celdas.
2. Dirección (Address): La dirección de memoria se utiliza para seleccionar una palabra específica (fila) dentro de la matriz. Un decodificador de direcciones recibe la dirección de entrada y activa la línea de fila correspondiente.
3. Acceso Aleatorio (Random Access): La característica principal de las memorias semiconductoras (RAM y ROM) es su acceso aleatorio. Esto significa que el tiempo que tarda el procesador en acceder a cualquier celda de memoria es prácticamente el mismo, independientemente de su ubicación física, lo que contrasta con el acceso secuencial de medios como las cintas magnéticas o incluso los discos duros (que requieren movimiento mecánico, es decir, deben recorrer todo la memoria hasta llegar a la ubicación requerida).

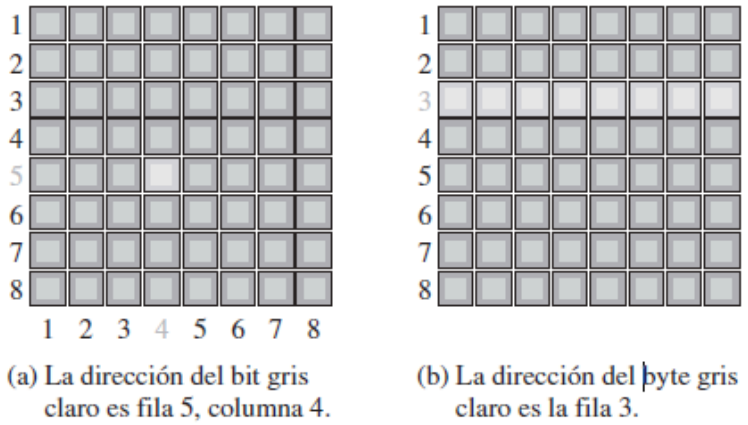

Figura 1.1 - Direcciones de memoria de una matriz bidimensional

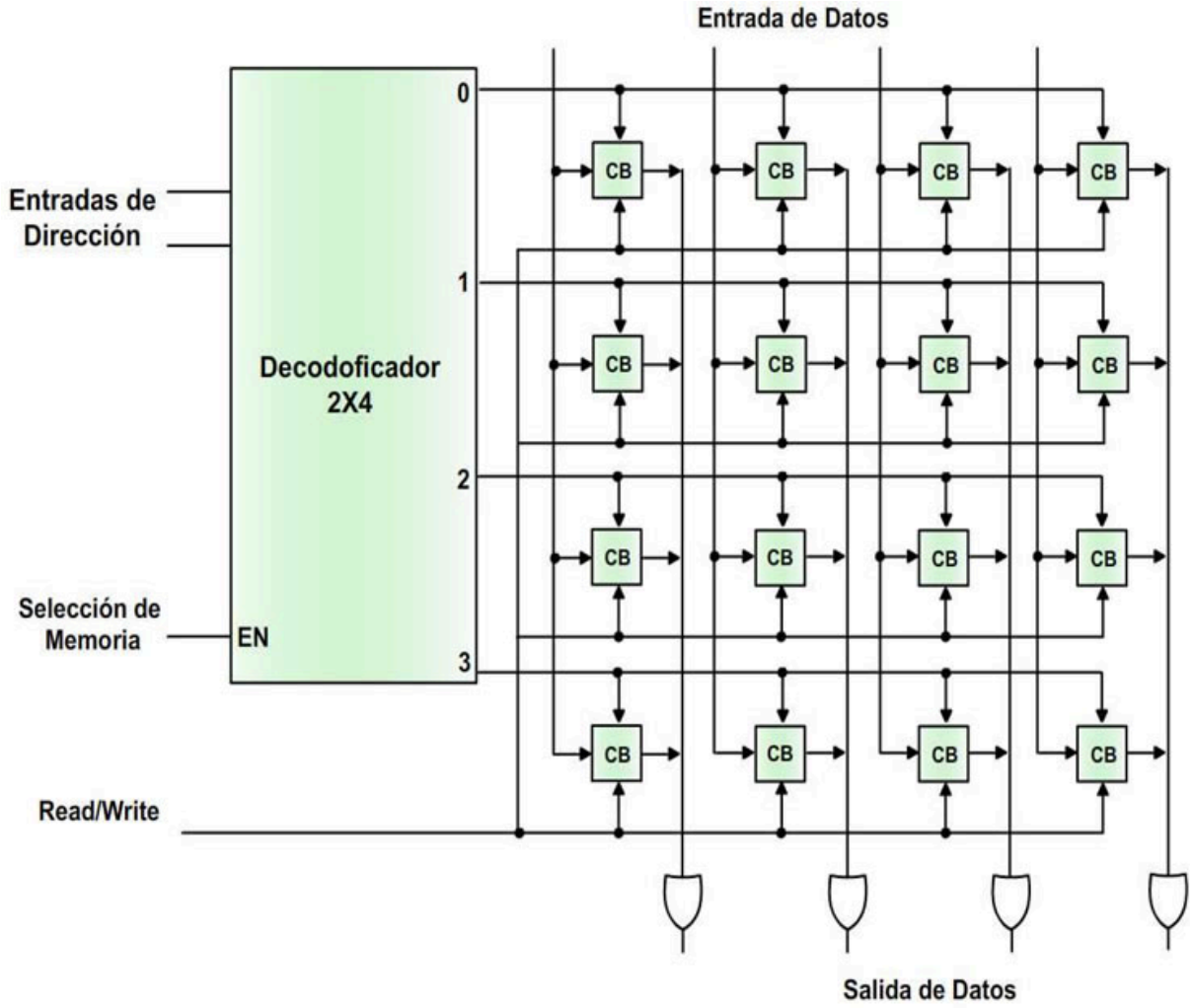

Figura 1.2 - Diagrama de bloques de una celda de memoria. con Entradas de dirección, Selección de memoria y decodificador de la palabra según la selección.

Clasificaciones de la celda de memoria

La tecnología de la celda define el tipo de memoria y sus características.

Su clasificación principal es sobre su Volatilidad, si retienen o pierden sus datos al cortar el suministro eléctrico.(Harris & Harris, 2007)

Memorias No Volátiles (ROM y Derivadas)

Estas memorias conservan su contenido de forma indefinida, incluso sin suministro eléctrico.

1. ROM (Read Only Memory)
    - Mecanismo de Almacenamiento: El bit se almacena como la presencia o ausencia de un transistor en la celda.
    - Programación: El contenido se especifica durante la fabricación (en la máscara) y no puede ser alterado posteriormente.
    - Uso típico en Firmwares y datos inmutables en productos de consumo masivo.
2. PROM (Programmable Read Only Memory)
    - Mecanismo de Almacenamiento: Inicialmente tiene un transistor en cada celda.
    - Programación: El usuario programa aplicando un alto voltaje que quema (volatiliza) selectivamente fusibles internos.
    - Tipo: Es de una sola programación (one-time programable), ya que los fusibles no se pueden reparar.
    - Uso típico en Programas permanentes personalizados.
3. EPROM (Erasable Programmable Read Only Memory)
    - Mecanismo de Almacenamiento: Utiliza un transistor de puerta flotante. La programación se realiza inyectando electrones en la puerta.
    - Borrado: El borrado es reversible y se logra exponiendo el chip a luz ultravioleta (UV) intensa durante aproximadamente media hora para eliminar los electrones atrapados.
    - Uso típico en Prototipos y sistemas que requieren actualizaciones ocasionales.
4. EEPROM (Electrically Erasable PROM) y Flash
    - Mecanismo de Almacenamiento: Similar al EPROM, usando un mecanismo reversible para el borrado.
    - Borrado: Incluyen circuitos en el chip para borrar la memoria, eliminando la necesidad de luz UV.
    - Diferencia: EEPROM borra y reescribe bytes individuales. Flash borra bloques más grandes de bits, lo que la hace más económica y popular para el almacenamiento de gran capacidad.
    - Uso típico: Para EEPROM:Almacenamiento de configuraciones de sistemas (Ej BIOS) o parámetros que cambian con poca frecuencia; Para Flash: Almacenamiento masivo (Ej: SSD, pendrives, tarjetas SD). Una memoria Flash ofrece una mejor combinación de densidad y velocidad de borrado/escritura de bloques.

Memorias Volátiles (RAM - Random Access Memory)

Estas memorias pierden su contenido cuando se interrumpe la alimentación.

1. SRAM (Static Random Access Memory)
    - Mecanismo de Almacenamiento (Celda): Es estática porque no necesita refresco para mantener los datos. Almacena un bit utilizando un par de inversores acoplados (cross-coupled inverters).(Harris & Harris, 2007)

- Implementación y Características:
    - La celda de SRAM es más compleja, típicamente requiere unos seis transistores por bit.
    - Es una memoria más rápida (tiene menor latencia) que la DRAM.
    - Se utiliza para construir registros internos (register files) y cachés en el mismo chip que el procesador. (Dally et al., 2016)
- Uso Principal: Memorias Caché del CPU y Conjuntos de Registros de un procesador

2. DRAM (Dynamic Random Access Memory)
- Mecanismo de Almacenamiento (Celda): Es dinámica porque su celda almacena un bit como una carga (presencia o ausencia) en un capacitor. El acceso al capacitor se controla mediante un único transistor nMOS que actúa como interruptor.(Harris & Harris, 2007)
- Volatilidad y Acceso:
    - La carga en el capacitor se fuga gradualmente, por lo que el contenido debe ser refrescado (leído y reescrito) cada pocos milisegundos.
    - La operación de lectura es destructiva, es decir, leer los datos elimina la carga, por lo que la palabra leída debe ser restituida inmediatamente.
- Implementación y Costo:
    - Requiere solo un transistor y un capacitor por celda.
    - Es más densa y, por lo tanto, tiene menor costo por bit que la SRAM.
    - Tiene mayor latencia (es más lenta) que la SRAM.
- Uso Principal: Memoria Principal del sistema(módulos DDR, la RAM que se inserta en la placa base).

Otra Clasificación a tener en cuenta es sobre su Capacidad de Modificación:
- Lectura/Escritura (R/W): Permiten tanto leer como escribir (RAM, Flash, EEPROM).
- Solo Lectura (RO): Están diseñadas para ser leídas (ROM, PROM).

Aplicaciones de Arreglos de Memoria en Sistemas de Cómputo

Los diferentes tipos de arreglos de memoria se utilizan jerárquicamente en un sistema de cómputo para equilibrar velocidad, capacidad y costo:

- Conjuntos de Registros (Register Files):
    - Es la forma más rápida y pequeña de almacenamiento de memoria de un sistema de computo.
    - Función: Almacenan variables temporales para acceso muy rápido (requerido por las instrucciones del procesador).
    - Tecnología: Se construyen generalmente con SRAM de múltiples puertos (multi ported SRAM) (para permitir múltiples lecturas y escrituras simultáneas) o, para muy pocos bits, con Flip-flops.
    - Es más rápido que el caché. requiere múltiples puertos de lectura y escritura para permitir que la unidad de control lea y escriba varios datos en paralelo durante la ejecución de una instrucción.
    - Capacidad típica de decenas de bytes.

- Memoria Caché (Cache Memory):
    - Actúa como amortiguador de alta velocidad para la CPU y la memoria principal (DRAM).
    - Función: Almacena datos e instrucciones de uso frecuente para reducir la latencia de acceso a la memoria principal.
    - Tecnología: Se construye con SRAM debido a su velocidad, a pesar de su mayor costo y menor capacidad en comparación con DRAM.
    - Es transparente para el programador. La gestión de que datos entran y salen se realiza por hardware mediante políticas de mapeo y reemplazo.
    - Capacidad típica: KiloBytes - MegaBytes.

- Memoria Principal (Main Memory):
    - Es el espacio de trabajo del sistema.
    - Función: Almacena los programas y datos en ejecución del sistema.
    - Tecnología: Principalmente DRAM, por su bajo costo y alta densidad, a pesar de ser más lenta que la SRAM.
    - Capacidad típica: GigaBytes.

- Memoria de Almacenamiento (Storage/Non-Volatile Memory):
    - Esta memoria se encuentra fuera de la jerarquía de accesos rápidos, pero es crucial para la persistencia.
    - Función: Almacenamiento persistente de datos (archivos, sistema operativo, etc.).
    - Tecnología: Flash (en dispositivos modernos como SSD) y ROM (para firmware y código de inicio).
    - Capacidad típica de TeraBytes.

Especificaciones de la actividad

La actividad consiste en la descripción de hardware de algunos componentes de memoria.

- Una ROM(Memoria de Solo Lectura) de 512 x 32 bits
- Una RAM(Memoria de Acceso Aleatorio) de 512 x 32 bits, de dos puertos uno para lectura y uno para escritura, el de escritura tiene una máscara de escritura para cada bloque de 8 bits.
- Un Registro de 32x32 bits, con tres puertos, dos de lectura y uno de escritura.
- Una RAM de 16x4 bit, para implementar en una FPGA(este sería la parte de laboratorio de la actividad).

Además de realizar la descripción de hardware, se realizaron bancos de prueba automáticos para cada componente.

Una ROM (Memoria de Solo Lectura) de 512 x 32 bits

Esta ROM debe leer una memoria al ingresar un vector de dirección y devolver una palabra de 32 bits.

Este cuenta con 2 entradas, una entrada de 9 bits la dirección(addr) y una entrada de reloj(clk), y tiene una salida de 32 bits (dout).

Una RAM (Memoria de Acceso Aleatorio) de 512 x 32 bits, de dos puertos uno para lectura y uno para escritura, el de escritura tiene una máscara de escritura para cada bloque de 8 bits.

Para esta RAM se puede ingresar a través de los vectores de dirección, a modo de leer o escribir algún dato en la memoria. En la escritura se tiene una máscara de escritura, donde se controla si se escribe o no cada bloque de 8 bits. En la salida se muestra la palabra que está observando la dirección de lectura.

Este cuenta con 5 entradas: dos entradas de direccion de 9 bits, una para leer(addr r) y una para escribir(addr_w), una entrada de dato(din), una de control de 4 bits para la entrada de escritura, como una máscara de escritura(we)”write enable” y una entrada de reloj (clk). Una salida de datos de 32 bits.

Un Registro de 32x32 bits, con tres puertos, dos de lectura y uno de escritura.

Esta memoria parte de un arreglo de 32 filas y 32 columnas, donde al recibir dos entradas de dirección o de fila para la lectura muestra las palabras de 32 bits o los 32 datos de esas filas en las 32 columnas. Además al recibir una dirección de escritura, y a través de una una máscara para cada bloque de 8 bits, se puede escribir en la dirección de escritura seleccionada.

Este cuenta con 6 entradas: tres entradas de dirección de 5 bits, dos para leer(addr_r1 y addr_r2) y una para escribir(addr_w), una entrada de dato(din), una de control de 4 bits para la entrada de escritura, como una máscara de escritura(we)”write enable” y una

entrada de reloj (clk). Dos salidas de 32 bits(dout _1 y dout_2) una para cada dirección de lectura.

Una RAM de 16x4 bit, para implementar en una FPGA

En esta memoria se puede ingresar un vector de dirección de 4 bits, donde se puede escribir un dato de entrada si la escritura se encuentra habilitada. En la salida se lee el dato a donde está apuntando la dirección.

Cuenta con 4 entradas: una entrada de 4 bits de dirección (addr), una de 4 bits de dato de entrada(din), una habilitación para la escritura(we) y una entrada de reloj. En la salida(dout) de 4 bits se lee el dato que marca la dirección.

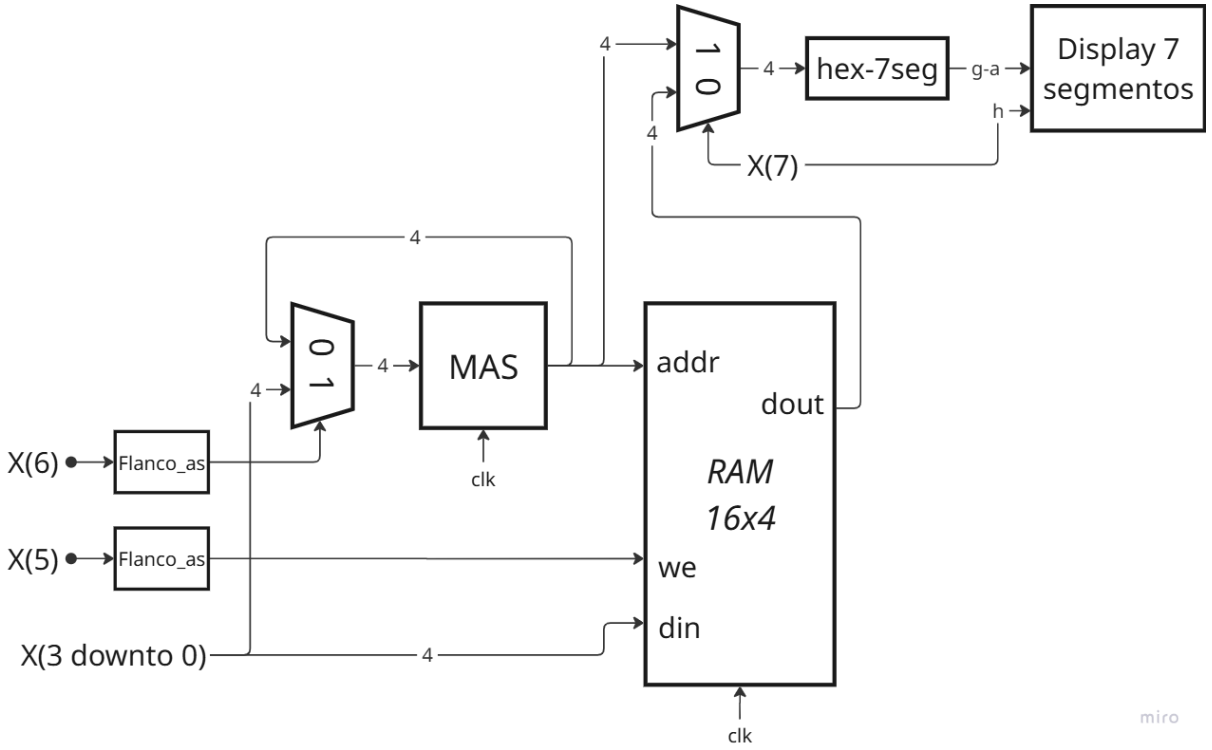

Figura 1.3 - Diagrama para implementación en FPGA de una RAM 16x4

En esta implementación se utilizan 8 interruptores y un display de 7 segmentos.

Para los interruptores X(7 a 0) se los dividió de la siguiente manera:

- X(3 a 0) es el ingreso de un vector de 4 bits, en este caso “din”.
- X(5) es la habilitación para la escritura sobre la memoria.
- X(6) es la habilitación para ingresar el vector de din como una dirección a la memoria.
- X(7) es un selector para lo que muestra el display de 7 segmentos, si está activo en el display se muestra la dirección guardada de la RAM en la MÁS, y si está en bajo muestra la salida (dout) de la RAM.

Además se agregó una memoria para la dirección, para que la misma quede guardada durante un tiempo indeterminado, hasta decidir cambiarla.

También un detector de flanco ascendente para las señales de X(5) y X(6), para evitar múltiple ingreso de datos por rebote.

Materiales y Métodos

Para esta actividad lo principal fue entender sobre los bloques de memoria, y la estructura de modo que la herramienta de síntesis reconozca el arreglo como memoria real (BRAM) y no como lógica combinacional.

Se deben definir las variables con las cuales debe trabajar la unidad de memoria, en general se usan las siguiente estructura y variables:

```
generic (
    constant init_file : string := ""
);
port (
    clk     : in  std_logic;
    we      : in  std_logic;
    addr    : in  std_logic_vector(8 downto 0);
    din     : in  std_logic_vector(7 downto 0);
    dout    : out std_logic_vector(7 downto 0)
);
```

Donde la constante init_file es un archivo vacío, lugar donde se inicializa la memoria, es decir todo su arreglo de direcciones y palabras está vacío.

Luego tenemos las variables:

- clk: señal de reloj, para que este sea sincronico
- we: write enable, la habilitación de la escritura
- addr: address. es la dirección a la cual se apunta.
- din: dato de entrada a la memoria, es el dato que se desea escribir
- dout: dato de salida de la memoria, es lo que se devuelve al leer la memoria.

También se utilizará un nuevo tipo de variable

```
type ram_type is array (511 downto 0) of std_logic_vector(7 downto 0);
```

Donde el nombre es genérico y es una variable de tipo arreglo(array) con,para una RAM de 512x8 de ejemplo, 512 filas o direcciones o palabras, y 8 columnas o largo de palabras.

Además a la hora de trabajar con una arreglo y como él mismo debe leer y/o escribir algo sobre algo, debe existir un lugar donde tenga el espacio para hacerlo, por lo que se define una función la cual cree un archivo o registro donde se pueda hacer todo eso antes de utilizarse, se utilizaron las siguientes sentencias:

```
impure function init_ram return ram_type is
    file ram_file : text;
    variable ram_data : ram_type := (others => (others => '0'));
```

```
    variable line_content : line;
    variable addr_index : integer := 0;
    variable valido : boolean;
    variable status : file_open_status;
  begin
    file_open(status, ram_file, init_file, read_mode);
    if status = open_ok then
      while not endfile(ram_file) loop
        readline(ram_file, line_content);
        hread(line_content, ram_data(addr_index), valido);
        if valido then
          addr_index := addr_index + 1;
        end if;
      end loop;
    end if;
    return ram_data;
  end function init_ram;

  signal ram : ram_type := init_ram;
```

Con ello tenemos inicializado una señal de tipo arreglo con la cual se puede trabajar, además se controla que exista y este sea válido para poder leer y/o escribir.

Continuamos con la sección combinación de la memoria, donde se puede leer y/o escribir.

```
  process(clk)
  begin
    if rising_edge(clk) then
      if we = '1' then
        ram(to_integer(unsigned(addr))) <= din;
      end if;
      dout <= ram(to_integer(unsigned(addr)));
    end if;
  end process;
```

Para ello se utiliza un proceso, donde en cada ciclo de reloj se puede:

- Si la máscara de escritura está activa se escribe el dato de entrada (din) en la dirección(addr) a la cual está apuntando de la memoria. Si la misma no está activa no se escribe nada.
- Se puede leer la salida(dout), pues cambia en cada ciclo con el dato al cual está apuntando la dirección(addr).

Resultados

Para esta parte de resultados se desarrollará un poco como se trabajó con cada uno de los componentes a desarrollar, pues todos parten de la misma base(bloque de memoria), pero con algunas pequeñas diferencias entre ellos.

Una ROM (Memoria de Solo Lectura) de 512 x 32 bits

Esta ROM debe leer una memoria al ingresar un vector de dirección y devolver una palabra de 32 bits.

Para este caso al ser una memoria de solo lectura, no son necesarias las entradas de din(dato de entrada) y de we (habilitación de escritura). por lo que es el componente más simple a realizar.

Es decir la única función de este componente es, una vez dada una dirección (addr), lee la palabra de dicha dirección en la salida(dout).

Una RAM (Memoria de Acceso Aleatorio) de 512 x 32 bits, de dos puertos uno para lectura y uno para escritura, el de escritura tiene una máscara de escritura para cada bloque de 8 bits.

Para este caso tenemos una memoria con lectura y escritura, pero tiene dos entradas de direcciones separadas.

Por lo que este componente lee el valor de la dirección de lectura(addr_r) a la vez que puede escribir en otra dirección(addr_w). Notar que si se escribe a la misma vez que se lee se verán datos distintos, no el dato nuevo, pues el guardado del nuevo dato se hace un ciclo de reloj atrasado al ser un proceso sincrónico.

Además cuenta con una máscara de entrada, el vector we de 4 bits, donde cada bit habilita la escritura para cada bloque de 8 bits de cada palabra de 32 bits. Por lo que en cada ciclo se verifica primero la máscara de escritura antes de escribir el dato de entrada.

Un Registro de 32x32 bits, con tres puertos, dos de lectura y uno de escritura.

Para este componente ya hay otras diferencias, pues ahora se trata de un registro no una RAM o ROM como se venía tratando, por lo que se se facilita la inicialización del mismo, simplemente haciendo:

```
signal ram : ram_type := (others => (others => '0'));
```

Como no es necesario inicializar desde un archivo externo, no es necesaria tampoco la utilización de la función para la verificación de la existencia del mismo. Esto simplifica un poco la descripción general.

Ahora en cuanto a su funcionamiento y entradas y salidas, esta vez se tiene 3 entradas importantes pues ahora hay 2 de lectura y una de escritura, la escritura es igual que en caso

anterior con un máscara de entrada. Pero al contar con dos entradas de lectura también debe haber dos salidas, pues dichas lecturas se pueden realizar en distintas direcciones. Esto agrega una salida más a nuestra memoria pero no es necesario modificar mucho más.

Una RAM de 16x4 bit, para implementar en una FPGA

Esta RAM era relativamente simple, pues solo cuenta con su inicialización, y luego escritura y lectura a través de una sola dirección, para la lectura tiene una la habilitación(we)

Para la implementación en la FPGA se hizo según referencia de la Figura 1.3, utilizando una pequeña memoria para guardar la dirección seleccionada, así como detectores de flanco como máquinas subordinadas para facilitar un poco la descripción. También un decodificador de hexadecimal a 7 segmentos, para tener una salida del display de 7 segmentos.

Conclusiones

Con esta actividad se puede terminar de comprender el principal objetivo de la descripción de hardware y sus estructuras.

La descripción de hardware no es simplemente escribir código, sino especificar de forma estructural y comportamental cómo debe funcionar un circuito digital. Al seguir las plantillas recomendadas por el fabricante y adoptar estructuras coherentes con la arquitectura del dispositivo, el sintetizador puede inferir componentes físicos adecuados —como bloques BRAM en lugar de lógica combinacional— lo cual mejora significativamente el rendimiento, el consumo de recursos y la claridad del diseño.

El desarrollo de las distintas memorias (ROM, RAM dual-port, registro multi-puerto y RAM para FPGA) permitió observar diferencias fundamentales entre tecnologías, métodos de inicialización, estrategias de acceso y comportamientos sincrónicos.

Referencias

- Harris, D. M., & Harris, S. L. (2007). *Digital design and computer architecture*. Elsevier.
- Dally, W. J., Harting, R. C., & Aamodt, T. M. (2016). *Digital design using VHDL: A systems approach*. Cambridge University Press.
- Arquitectura37. (2014). *Memorias*. https://arquitectura37.webnode.com.co/novedades/memorias/
- Instituto Consorcio Clavijero. (s. f.). *Sistemas Digitales II – 1.6 Memorias de acceso aleatorio*. https://cursos.clavijero.edu.mx/cursos/072_sdII/modulo1/contenidos/tema1.6.html