INSTITUTO POLITÉCTICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ESCOM

Carrera:

Ingeniería en sistemas computacionales

Unidad de Aprendizaje:

Sistemas en chip

Entregable 1:

Práctica de segmentación - Verilog

Integrantes:

Cazares Cruz Jeremy Sajid

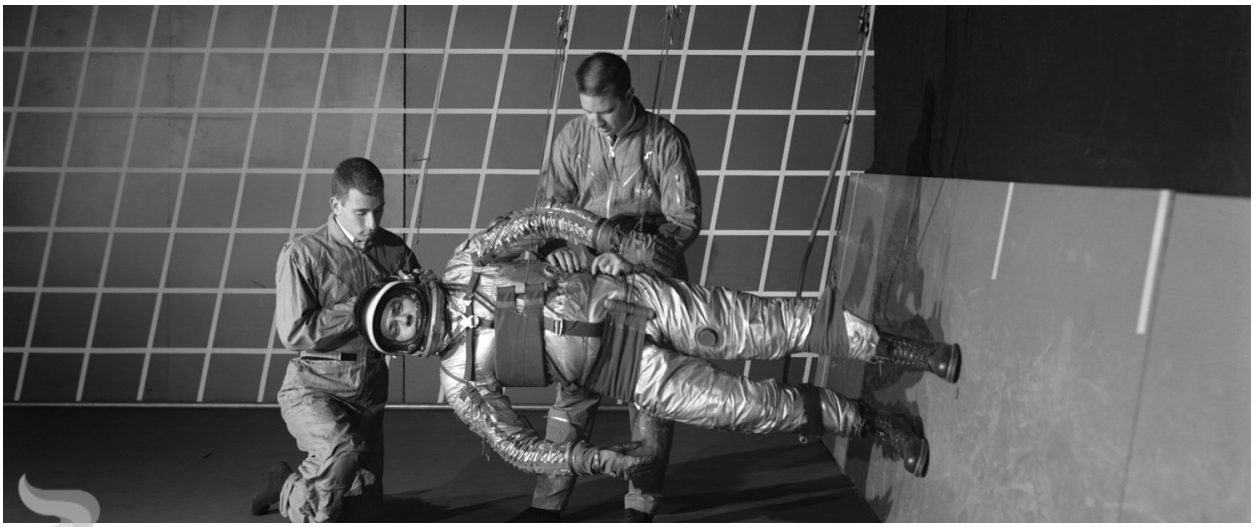
Bucio Barrera Oscar Daniel

Perez Ortiz Saúl

Acosta Cortes Gerardo

Fecha de entrega

18 de septiembre de 2023



Reporte Práctica 1

Descripción General:

El módulo descrito es una implementación de una memoria caché simple. Utiliza una estructura de conjunto directo, donde cada dirección de memoria se asocia con una única ubicación en la caché. Se proporcionan mecanismos para escribir y leer datos en la memoria caché, así como para comprobar si un dato está presente en la caché (a través de una señal de "acceso exitoso" o "hit").

Descripción Detallada:

1. **Definición del módulo y parámetros:**

```
module memoria #(
    parameter ADDR = 32,
    parameter   BOFF = $clog2(4),
    parameter IDX = 5,
    parameter   TAG = ADDR - (BOFF + IDX),
    parameter   DATO = 32
)
```

   - El módulo se llama `memoria`.

- Se define un conjunto de parámetros:
  - `ADDR`: Representa el número de bits de la dirección completa (por defecto es 32 bits).
  - `BOFF`: Es el número de bits utilizados para el desplazamiento de bloque. Es calculado como el logaritmo en base 2 de 4 (que es 2). Esencialmente, determina la cantidad de palabras que puede almacenar un bloque de memoria.
  - `IDX`: Es el número de bits utilizados para indexar la memoria caché. Define el número de entradas o líneas que tiene la caché.
  - `TAG`: Es el número de bits utilizados para el etiquetado. Se calcula restando `BOFF` y `IDX` de `ADDR`.
  - `DATO`: Representa el número de bits de los datos que se van a almacenar (por defecto es 32 bits).

2. **Lista de puertos del módulo:**

```
(
    input           clk_i,
    input           readen_i,
    input   [ADDR-1:0]  addr_i,
    input   [DATO-1:0] dato_i,
    input           writeen_i,
    output  [DATO-1:0]  dato_o,
    output          hit_o
)
```

- `clk_i`: Es la entrada de la señal de reloj.
- `readen_i`: Señal que indica cuando se quiere leer de la memoria.
- `addr_i`: Dirección de memoria de donde se quiere leer o escribir.
- `dato_i`: Datos que se quieren escribir en la memoria.
- `writeen_i`: Señal que indica cuando se quiere escribir en la memoria.
- `dato_o`: Datos leídos de la memoria.
- `hit_o`: Señal que indica si los datos requeridos estaban en la memoria caché (acceso exitoso).

3. **Sección de declaración de señales:**

```
wire  [IDX-1:0]   index;
wire  [TAG-1:0]    tag;
reg         validmem;
reg [TAG-1:0] tagmem;
reg [DATO-1:0]  datamem;
reg         tagvalido;
```

- `index`: Señal que almacena el índice extraído de la dirección dada. Se utiliza para indexar la memoria caché.
- `tag`: Señal que almacena el valor de etiqueta extraído de la dirección dada.
- `validmem`: Indica si los datos en el índice actual de la memoria caché son válidos o no.
- `tagmem`: Almacena el valor de etiqueta de la entrada actual de la memoria caché.
- `datamem`: Almacena el valor de datos de la entrada actual de la memoria caché.
- `tagvalido`: Indica si el valor de etiqueta dado coincide con el valor de etiqueta almacenado en la memoria caché.

4. **Declaración de memorias:**

```
reg   [2**IDX-1:0]   memoriavalid;
reg     [TAG-1:0]    memoriatag  [2**IDX-1:0];
reg   [DATO-1:0]   memoriadata [2**IDX-1:0];
```

- `memoriavalid`: Es una memoria que indica si una entrada particular de la memoria caché es válida o no. Si es 1, significa que los datos en esa entrada son válidos; si es 0, los datos no son válidos.
- `memoriatag`: Es una memoria que almacena el valor de etiqueta para cada entrada de la memoria caché.
- `memoriadata`: Es una memoria que almacena los datos reales para cada entrada de la memoria caché.

5. **Extracción de índice y etiqueta:**

```
assign index = addr_i[IDX+BOFF-1:BOFF];
assign  tag =  addr_i[ADDR-1:IDX+BOFF];
```

- Las direcciones se dividen en tres partes: etiqueta (`tag`), índice (`index`) y desplazamiento de bloque. Estas asignaciones extraen el índice y la etiqueta de la dirección completa.

6. **Puerto de escritura:**

```
always @(posedge clk_i)
begin
    if (writeen_i)
    begin
     memoriavalid [index] = 1'b1;
     memoriatag [index]   = tag;
     memoriadata [index]  = dato_i;
    end
end
```

- Este bloque describe cómo se escriben los datos en la memoria caché. Cuando `writeen_i` es activo (alto), los datos se escriben en la memoria caché en la entrada correspondiente al índice extraído. Además, se establece la entrada correspondiente en `memoriavalid` a 1, indicando que los datos en esa entrada ahora son válidos.

7. **Puerto de lectura:**

```
always @(*)
begin
    if (readen_i)
    begin
     validmem = memoriavalid [index];
     tagmem = memoriatag [index];
     datamem = memoriadata [index];
    end
    else
    begin
     validmem = 1'b0;
     tagmem = {TAG{1'b0}};
     datamem = {DATO{1'b0}};
    end
end
```

- Cuando `readen_i` es activo este bloque lee los datos de la memoria caché. Si los datos en esa entrada son válidos y la etiqueta coincide, entonces es un acceso exitoso (`hit`). Si `readen_i` no es activo, se asignan valores predeterminados a `validmem`, `tagmem`, y `datamem`.

1. **Comparación de etiquetas:**

```
always @(*)
begin
    if  (tag == tagmem)
     tagvalido = 1'b1;
    else
     tagvalido = 1'b0;
end
```

   - Este bloque compara la etiqueta extraída de la dirección dada con la etiqueta almacenada en la entrada correspondiente de la memoria caché. Si coinciden, `tagvalido` se establece en 1, indicando un acceso exitoso.

2. **Generación de señales de salida:**

```
assign hit_o = validmem & tagvalido;
assign dato_o = (hit_o) ? datamem : {DATO{1'b0}};
```

   - `hit_o` se establece en 1 si los datos en la entrada actual son válidos y la etiqueta coincide. Es decir, indica un acceso exitoso.
   - `dato_o` devuelve los datos leídos de la memoria caché si hay un acceso exitoso; de lo contrario, devuelve un valor predeterminado (todos los bits en 0).

3. **Fin del módulo:**

```
endmodule
```

- Indica el final del módulo `memoria`.
