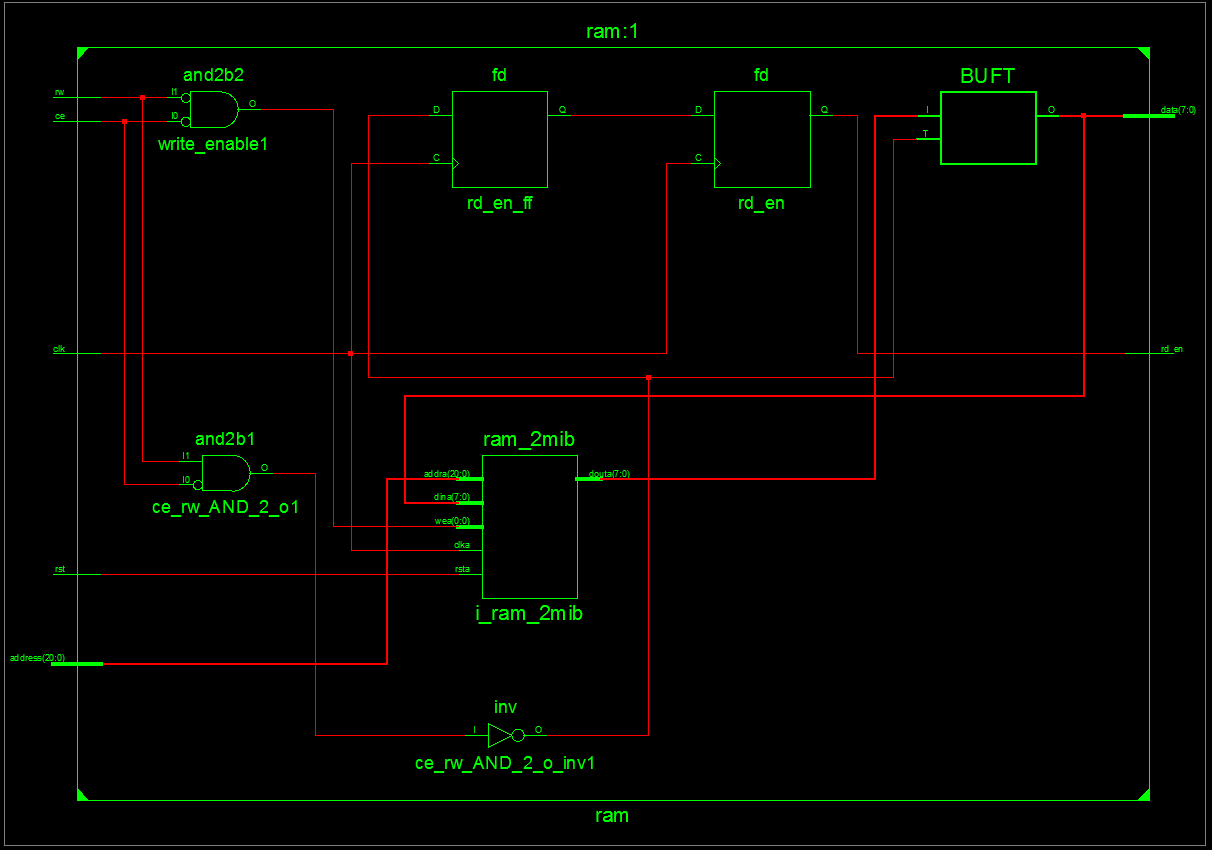
# Progreso

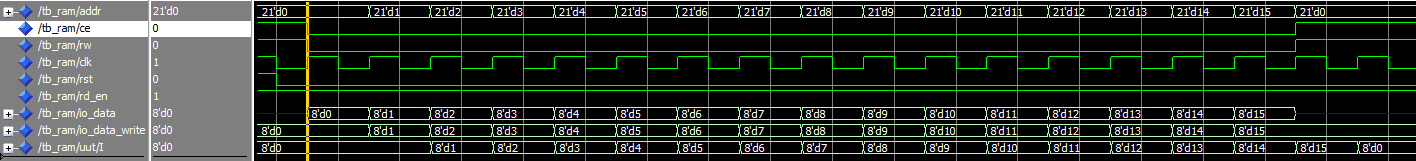
1 – Repositorio GitHub

2 – Módulo RAM: Contiene un componente Block Memory Single Port obtenido mediante CORE Generator, de 2MiB, más eficiente que crear un array de vectores. Este componente es conectado a los puertos del bloque contenedor RAM usando lógica que permita adaptar el bus de entrada-salida a los buses de data in y data out de la memoria. La herramienta de Xilinx infiere un buffer triestado a partir de la lógica descrita, como se aprecia en la imagen:

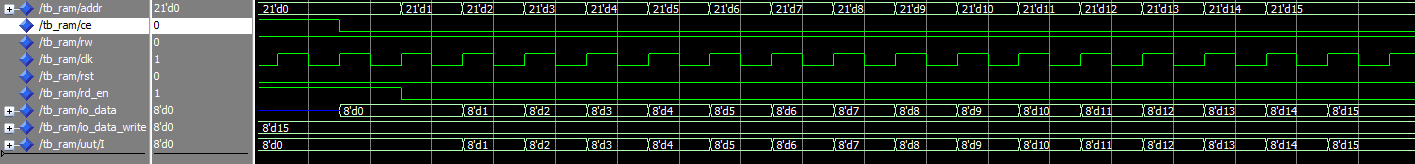


Para comprobar su correcto funcionamiento se ha implementado un testbench sencillo que compruebe la capacidad de escritura y lectura del módulo sobre la Block RAM. En las siguientes imágenes se puede apreciar el que el comportamiento es el esperado:

Escritura: En cada flanco de subida de reloj se escribe sobre la posición indicada por la dirección el dato que se encuentra en io\_data, que previamente se ha cargado en io\_data\_write

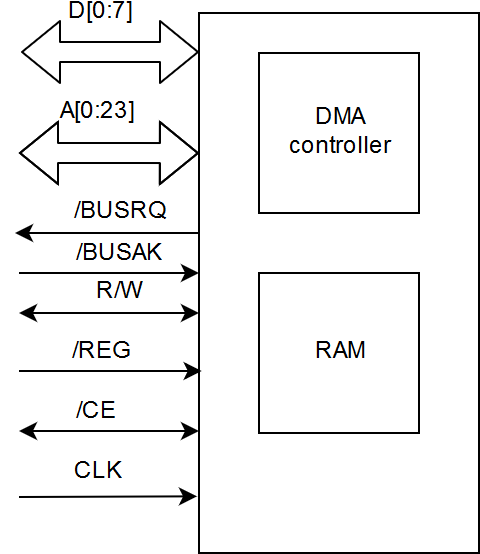


Lectura: Como se aprecia, la memoria tiene una latencia de 1 ciclo de reloj para proporcionar el dato de la posición que se le indique con la dirección v\_addr. Para indicar que el dato presente en el puerto data es el dato leído de la RAM, se utiliza la señal rd\_en, que se activa a nivel bajo cuando el dato se obtiene de la memoria tras esperar el periodo de latencia



2 – Modulo selección de direcciones: la memoria tiene capacidad para 2MiB = 221B -> Hacen falta 21 bits para direccionar todas las posiciones de memoria. Las direcciones tienen 24 bits, por lo que sobran los 3 MSB. El sistema tiene 12 registros < 24. Son necesarios 4 bits para crear direcciones que permitan acceder a todos los registros. Estas direcciones se crean usando los 4 LSB del bus de direcciones. Necesario distinguir cuando el

Diseñe el sistema indicado en la figura, que dispone de un bloque de memoria de 2 MiB, y un controlador de DMA con capacidad “Bus Master”.



El sistema se conecta al bus paralelo de un microprocesador, y deberá funcionar por defecto (tras el reset) como esclavo, sirviendo las operaciones de lectura y escritura del bus, en las direcciones indicadas. Bajo petición, podrá funcionar como maestro del bus, invirtiendo el rol, y realizando operaciones de lectura o escritura directamente.

La descripción del interfaz es la siguiente (los terminales precedidos por ‘/’ son activos a nivel bajo, las entradas/salidas son descritas desde el punto de vista del sistema a diseñar):

* D[0:7]: Bus de datos bidireccional de 8 bits.
* A[0:23]: Bus de direcciones de 24 bits (esclavo: entrada, maestro: salida). Nótese que al ser la memoria interna de 2 MiB, no usa el bus al completo.
* /CE: Chip Enable. Habilita las operaciones de bus (esclavo: entrada, maestro: salida).
* R/W: Lectura/escritura: Lectura si nivel alto, escritura si nivel bajo (esclavo: entrada, maestro: salida).
* /BUSRQ: Petición de bus (salida).
* /BUSAK: Autorización de maestro del bus (entrada).
* /REG: Acceso a los registros internos del sistema (entrada).
* CLK: Reloj del sistema. Los accesos a memoria se sincronizan con los flancos ascendentes de este reloj (entrada).

El sistema dispone de 12 registros de 8 bits, que son accedidos cuando /REG y /CE están activos, y direccionados por los 4 bits de menos peso de la línea de direcciones:

|  |  |  |
| --- | --- | --- |
| Dirección | Nombre | Función |
| 0 | BASEM | 16 bits de más peso de la dirección base en que se mapean los 2 MiB disponibles. Por defecto se asigna a la dirección 0, pero puede ser remapeado escribiendo en este registro. |
| 1 | BASEH |
| 2 | SRCL | Dirección origen SRC (byte bajo, medio y alto) |
| 3 | SRCM |
| 4 | SRCH |
| 5 | DSTL | Dirección destino DST (byte bajo, medio y alto) |
| 6 | DSTM |
| 7 | DSTH |
| 8 | LENL | Tamaño de transferencia LEN (byte bajo, medio y alto) |
| 9 | LENH |
| 10 | LENU |
| 11 | CTRL | Registro de control |

El registro de control tendrá los siguientes campos:

* Bit 0: START: Poner a 1 para iniciar la transferencia
* Bit 1: READ/WRITE: Si 0, el sistema leerá del bus paralelo en la dirección SRC, y lo almacenará en el bus interno en DST. Si 1, el sistema leerá de la dirección interna DST, y almacenará en el bus interno en DST.
* BITS 2-4: Auto incrementar/decrementar SRC con el valor en estos bits (interpretar como complemento a 1) tras cada copia.
* BIT 5-7: Auto incrementar/decrementar DST con el valor en estos bits (interpretarlos como complemento a 1) tras cada copia.

Para que el sistema funcione como maestro del bus, se deberán programar los registros SRC, DST y LEN con los valores deseados. Luego se deberá programar CTRL, poniendo el bit 0 a 1 (y el resto a los valores deseados).

Codifique el sistema diseñado en VHDL. Añada si lo considera necesario las líneas de control que considere. Codifique también el test-bench que permita verificar el funcionamiento del sistema, y ejecute la simulación.

Elabore un informe de las tareas realizadas describiendo tanto el diseño de la aplicación como del test-bench y un resumen de las pruebas realizadas y su resultado.