ARCHIVO DE REGISTROS.

Esta sección del procesador esta formada por 16 registros de 16 bits cada uno, los cuales llamaremos como R0, R1, … ,R15. Estos registros son los que contienen los datos que usan las instrucciones.

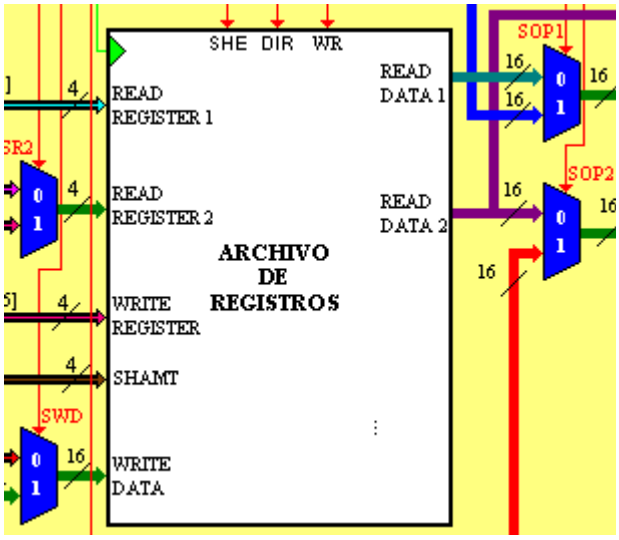

Ilustración 1 Archivo de registros.

Como se puede observar en la ilustración 1 el archivo de registros maneja las siguientes señales:

➢**Read Register 1**. Este es un bus de 4 bits de entrada. Con este bus se especifica uno de los 16 registros a usarse como primer operando en las instrucciones tipo R o tipo I. El dato del registro seleccionado con este bus será colocado en el bus de salida **Read Data 1.**

➢**Read Register 2**. Este es un bus de 4 bits de entrada, en el cual se especifica uno de los 16 registros a usarse como segundo operando en las instrucciones tipo R o tipo I. El dato del registro seleccionado con este bus será colocado en el bus de salida **Read Data 2.**

➢**Write Register**. Este es un bus de 4 bits de entrada, en el cual se especifica uno de los 16 registros donde se colocará el resultado en las instrucciones tipo R o tipo I. El dato colocado en el bus **Write Data** será almacenado en el registro seleccionado con este bus**.**

➢**Shamt**. Este es un bus de 4 bits de entrada, en el cual se especifica la cantidad de bits a recorrer en las instrucciones de corrimiento SLL y SRL. La cantidad de bits a recorrer puede ser 0, 1, 2, …, 15 bits ya sea a la izquierda o la derecha dependiendo de la instrucción.

➢**Write Data**. Este es un bus de 16 bits de entrada, en el cual se especifica el dato de 16 bits a cargar en el registro indicado por el bus **Write Register**.

Autor: Victor Hugo García Ortega

➢**Read Data 1**. .Este es un bus de 16 bits de salida, del cual se obtiene el dato contenido en el registro indicado por el bus **Read Register 1**.

➢**Read Data 2**. .Este es un bus de 16 bits de salida, del cual se obtiene el dato contenido en el registro indicado por el bus **Read Register 2**.

➢**CLK**. Señal de reloj. Las tareas del procesador se ejecutan en flanco de subida.

➢**SHE**. Señal de habilitación de corrimiento (Shift Enable). Cuando esta señal tiene 1 se puede realizar el corrimiento del registro indicado por el bus **Read Register 1**, el número de bits indicado por el bus **Shamt**. Esta señal se ejecuta de forma síncrona, es decir, se ejecuta en el momento que llega una flanco de subida de la señal de reloj.

➢**DIR**. Señal de dirección en el corrimiento. Cuando esta señal tiene 1 se realiza el corrimiento a la izquierda del registro indicado por el bus **Read Register 1**, el número de bits indicado por el bus **Shamt**. Cuando esta señal tiene 0 se realiza el corrimiento a la derecha del registro indicado por el bus **Read Register 1**, el número de bits indicado por el bus **Shamt**. Para que esta señal tenga efecto en el corrimiento se debe tener la señal SHE = 1. Esta señal se ejecuta de forma síncrona, es decir, se ejecuta en el momento que llega una flanco de subida de la señal de reloj.

➢**WR**. Señal de carga para el archivo de registro (Write). Cuando esta señal tiene 1 se realiza la carga del dato colocado en el bus **Write data** en el registro especificado por el bus **Write register**. Esta señal se ejecuta de forma síncrona, es decir, se ejecuta en el momento que llega una flanco de subida de la señal de reloj.

Arquitectura del archivo de registros.

Los registros R0, R1, … ,R15 se encuentran implementados en una memoria RAM de tres puertos: WRITE REGISTER, READ REGISTER1 y READ_REGISTER2. En el caso de los FPGA's de Xilinx los LUT's de los slices puede configurarse como una memoria Ram Distribuida multipuerto, por lo que se puede usar ese recurso dedicado para implementar el archivo de registros. La escritura en los registros se hace manera síncrona y la lectura se hace de manera asíncrona. La arquitectura del archivo de registros se muestra en la ilustración 2.

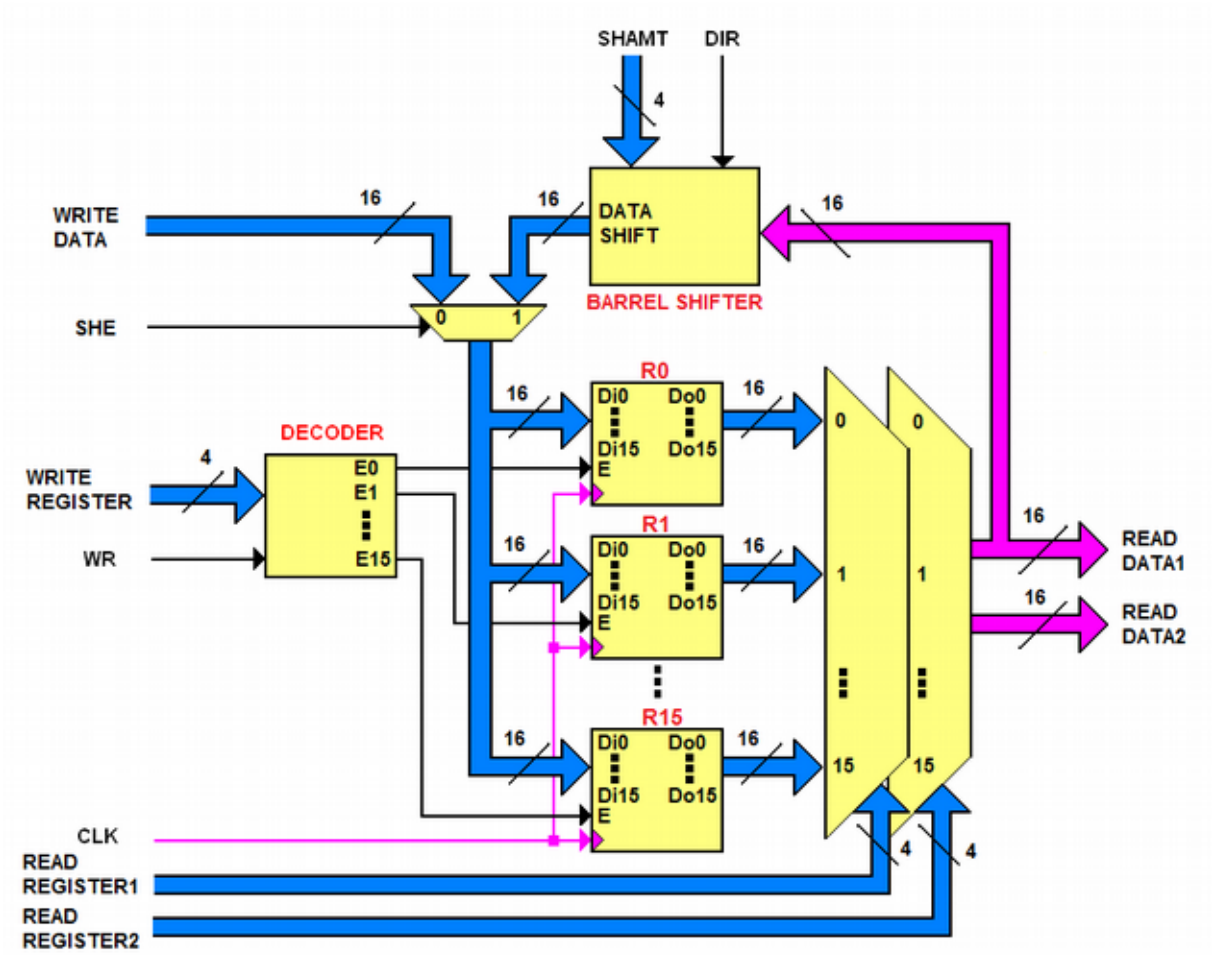

Ilustración 2: Arquitectura del archivo de registros

Autor: Victor Hugo García Ortega

Barrel Shifter.

La operaciones de corrimiento a la izquierda y a la derecha se realizan sobre el registro indicado por el bus **Read Register 1** y el resultado se coloca en el registro indicado por el bus **Write Register**. El número de bits a recorrer se indica por el bus **Shamt**. La señal **Shift Enable** (SHE) habilita la operación de corrimiento y la señal de **Dirección** (DIR) indica si el corrimiento es a la izquierda o a la derecha, tal como se muestra en la tabla 1. Estas operaciones de corrimiento implementan un módulo en el procesador llamado **Barrel Shifter**. Este módulo es el encargado de realizar corrimientos a la izquierda y a la derecha de "n" bits sobre alguno de los registros de 16 bits del archivo de registros en un solo ciclo de reloj.

SHE	DIR	Operación	Significado
1	0	Corrimiento a la derecha	REG[WriteRegister] = REG[ReadRegister1] >> Shamt
1	1	Corrimiento a la izquierda	REG[WriteRegister] = REG[ReadRegister1] << Shamt

Tabla 1 Funcionamiento del Barrel Shifter.

La arquitectura del Barrel Shifter esta formado por una configuración de multiplexores que permite realizar corrimientos a la izquierda y por otra que permite realizar corrimientos a la derecha. Esto se muestra en la ilustración 3. El resultado del corrimiento aplicado es seleccionado por el bit DIR.

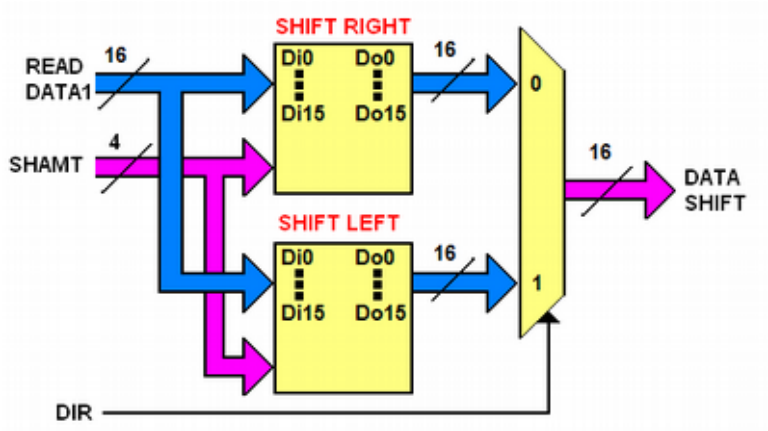

Ilustración 3: Barrel shifter

La arquitectura para corrimiento a la izquierda de 8 bits se muestra en la ilustración 4. En esta ilustración se observan 3 etapas de multiplexores donde se tienen las siguientes señales:

- DATAIN, es un bus de entrada de 8 bits, que indica es el número al cual se le va aplicar el

corrimiento.

- SHIFT, es un bus de entrada de 3 bits, que indica el número de bits que se va a recorrer a la izquierda el número colocado en DATAIN.
- DATAOUT, es un bus de salida de 8 bits, que indica el resultado del corrimiento aplicado a DATAIN.

En la arquitectura del archivo de registros la señal DATAIN es READ DATA1, SHIFT es SHAMT y DATAOUT es DATA SHIFT.
La primer etapa de multiplexores utiliza como selector SHIFT(0), por ser el bit 0, permite realizar el corrimiento de 0 bits si vale 0 y 2^0=1 bit si vale 1. Por esta razón, la salida de los multiplexores es:

SHIFT_DATA = DATAIN

SHIFT_DATA(J) = SHIFT_DATA(J), SHIFT(0) = 0
SHIFT_DATA(J-1), SHIFT(0) = 1, J-1 >= 0
0, SHIFT(0) = 1, J-1 < 0

La segunda etapa de multiplexores utiliza como selector SHIFT(1), por ser el bit 1, permite realizar el corrimiento de 0 bits si vale 0 y 2^1=2 bits si vale 1. Por esta razón, la salida de los multiplexores es:

SHIFT_DATA(J) = SHIFT_DATA(J), SHIFT(0) = 0
SHIFT_DATA(J-2), SHIFT(1) = 1, J-2 >= 0
0, SHIFT(1) = 1, J-2 < 0

La tercera etapa de multiplexores utiliza como selector SHIFT(2), por ser el bit 2, permite realizar el corrimiento de 0 bits si vale 0 y 2^2=4 bits si vale 1. Por esta razón, la salida de los multiplexores es:

SHIFT_DATA(J) = SHIFT_DATA(J), SHIFT(2) = 0
SHIFT_DATA(J-4), SHIFT(2) = 1, J-4 >= 0
0, SHIFT(2) = 1, J-4 < 0

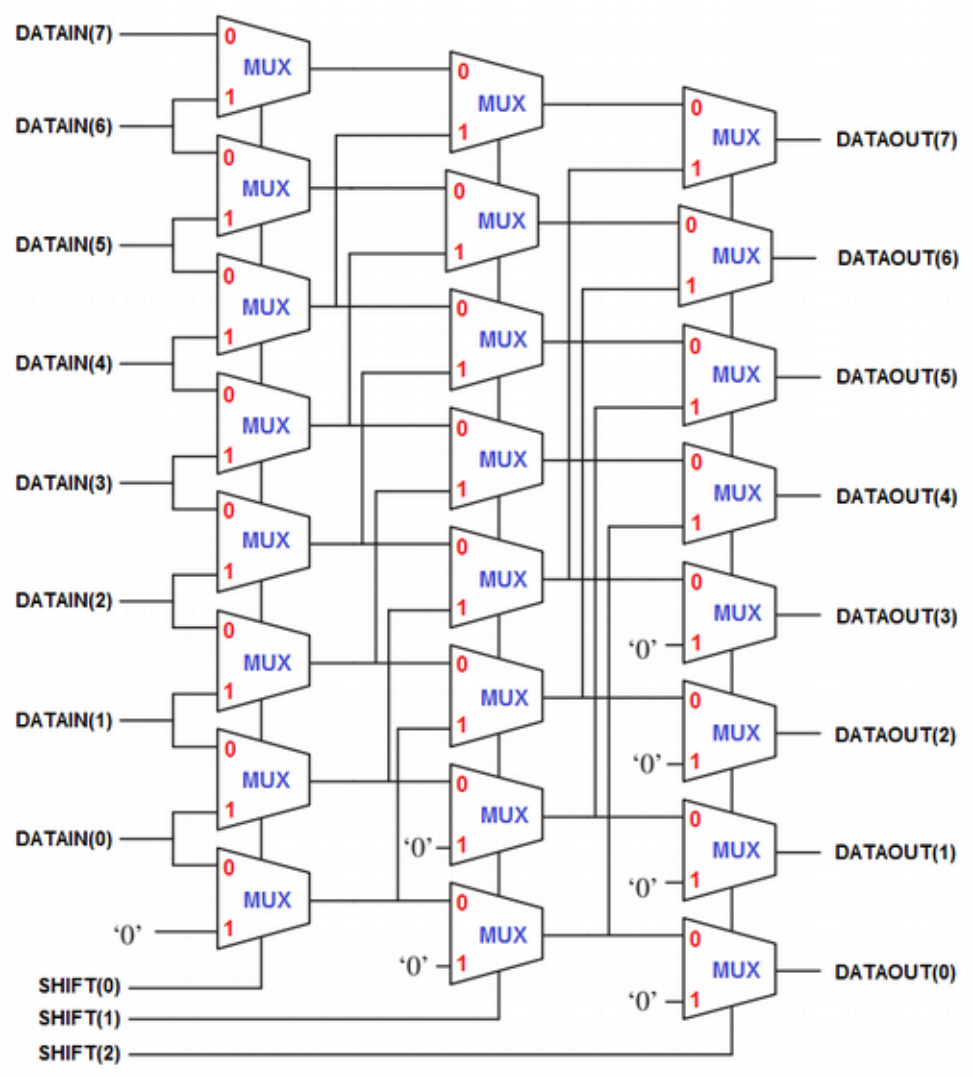

Ilustración 4: Barrel shifter para corrimiento a la izquierda de 8 bits

Autor: Victor Hugo García Ortega