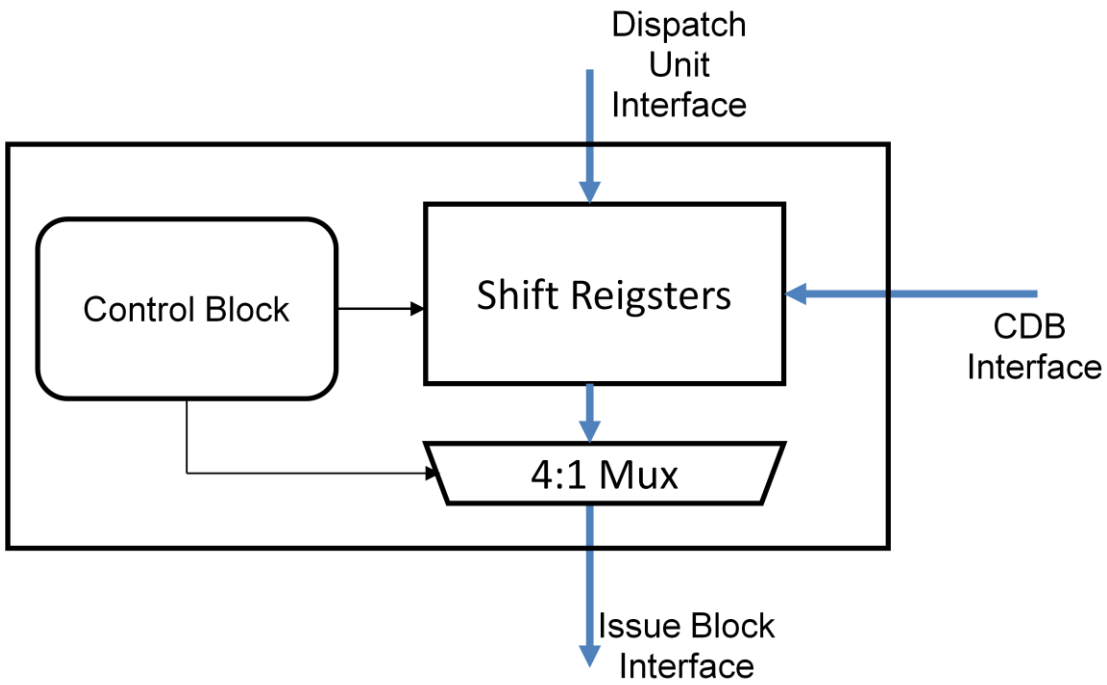
1 Introducción

La cola de ejecución de enteros tiene el mismo comportamiento que la cola de ejecución de multiplicaciones y divisiones. Más adelante se presentaran los detalles para que el mismo modulo pueda ser utilizado para los tres casos.

En las colas de ejecución las instrucciones esperan su turno para ser ejecutadas, que no necesariamente es de manera ordenada, es decir, la primera que entra no es la primera que sale. Una instrucción puede ser candidata a ser ejecutada una vez que todos sus operandos son conocidos. A las colas de ejecución pueden entrar instrucciones con uno o los dos operandos "desconocidos", es en estas colas donde las instrucciones actualizan el valor de sus operandos con la información publicada en el bus de datos común (CDB).

Las colas de ejecución para nuestra implementación únicamente tendrán 4 entradas e interactúan con la unidad de despacho, su respectiva unidad de ejecución, con la unidad de Issue y con el bus de datos común.



2 Especificaciones de Diseño.

2.1 Interfaces.

Interface con la unidad de despacho.

dispatch_rs_data: 32 bits de datos (rs)
dispatch_rs_tag: 5 bits de TAG (rs)
dispatch_rs_data_val: '1' indica que el dato es válido, '0' el dato es desconocido.
dispatch_rt_data: 32 bits de datos (rt)
dispatch_rt_tag: 5 bits de TAG (rt)
dispatch_rt_data_val: '1' indica que el dato es válido, '0' el dato es desconocido.
dispatch_opcode: 3 bits para seleccionar la operación de la ALU (solo usado para la cola de ejecución de enteros, para las otras colas vamos a "amarrar" este campo a "000".
Dispatch_shfamt: 5 bits del campo shift amount para una instrucción del tipo shift
dispatch_rd_tag : el TAG asignado para el registro destino.
dispatch_enable: '1' la unidad de despacho quiere escribir una nueva instrucción en la cola.
issueque_full: '1' la cola de ejecución se encuentra llena.

Interface con el bus de datos común (CDB).

cdb_tag: 5 bits de TAG,
cdb_data: 32 bits de datos.
cdb_valid: '1' indica que tanto el TAG y los datos son validos.

Interface con la unidad de ejecución.

issueque_ready: señal que indica a la unidad de issue que una instrucción esta lista.
issueque_rs_data : 32 bits de datos del operando rs
issueque_rt_data : 32 bits de datos del operando rt
issueque_rd_tag : 5 bits del tag del registro destino.
issueque_opcode : 3 bits del opcode para la ALU
issueque_shfamt: 5 bits del campo de shift amount
issueblk_issue : señal que indica que la instrucción lista ha sido ejecutada.

Interface con el retire bus.

Flush_valid: 1 bit de senial de control que indica que los datos almacenados en la cola tiene que ser eliminados.

2.2 Diseño de la cola de ejecución.

El diseño de la cola de ejecución tiene como principal reto el mantener un orden local, con respecto a otras instrucciones del mismo tipo, a pesar que las instrucciones pueden salir en diferente orden.

Cada entrada de la cola tiene los campos que se muestran en la siguiente figura. El campo de Rs_val y Rt_val se refiere a la validez del dato y no del tag, si dicho campo se encuentra en '1' quiere decir que el operando está "listo" y no tiene que ser actualizado por el CDB. El campo Valid se refiere a que la entrada de la cola se encuentra ocupada, este campo será usada para generar la bandera de "llena".

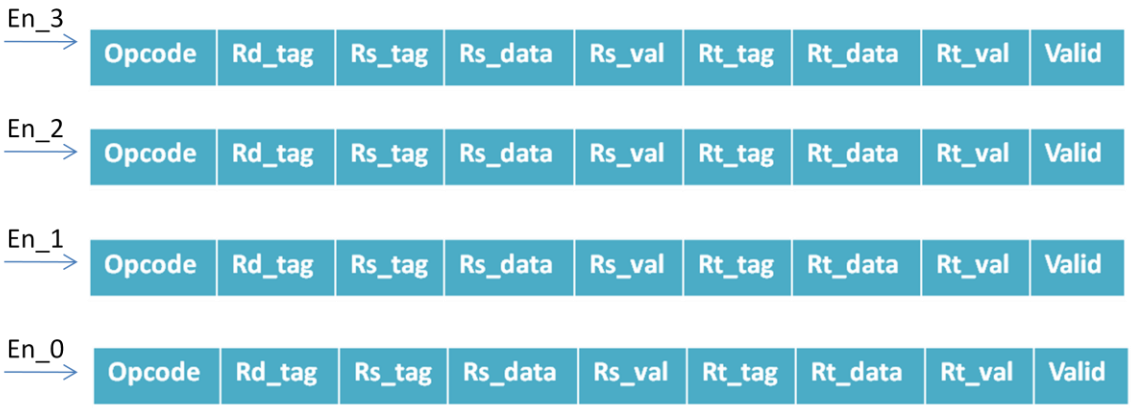


Figura 1.

Existen cuatro operaciones en esta cola, *Shift*, *Update, Add y flush.* Los Adds se dan cada vez que que entra una nueva instrucción a la cola. Los shifts se pueden generar ya sea por la entrada de una nueva instrucción o por la salida de una instrucción. Los updates se generan cuando el resultado de un TAG publicado en el CDB hace match con el tag (rs o rt) de una entrada en la cola. Los Flush se dan cuando el retire bus indica que las instrucciones pendientes en el backend tienen que ser eliminadas, para hacer flush simplemente el bit de valid de todas las entradas de la cola tiene que ser puesto en '0'.

Como se muestra en la figura 2 los únicos campos que necesitan un multiplexor para la operación de *shift* son los datos y el valid tanto de rs como de rt. Esto porque puede suceder que un *shift* al mismo tiempo que un *update*. El resto de los campos pasan directamente a la entrada de abajo. Las señales de En_[3:0] se encargan de habilitar cuales entradas de la cola tomaran nuevos valores. En el caso que solo se presente la operación de update se tiene que ser cuidadoso de no actualizar todos los campos de la entrada de la cola y solo actualizar los

datos y el valid del campo correspondiente. Existen dos maneras de asegurar esto; la primera consiste en tener una señal de enable independiente para estos campos (rs_data, rs_valid, rt_data, rt_val), la segunda consisten en mantener una única señal de enable para toda la entrada de la cola y poner multiplexores que retroalimenten los campos que no se actualizan con la operación de update como se muestra en la figura 3.

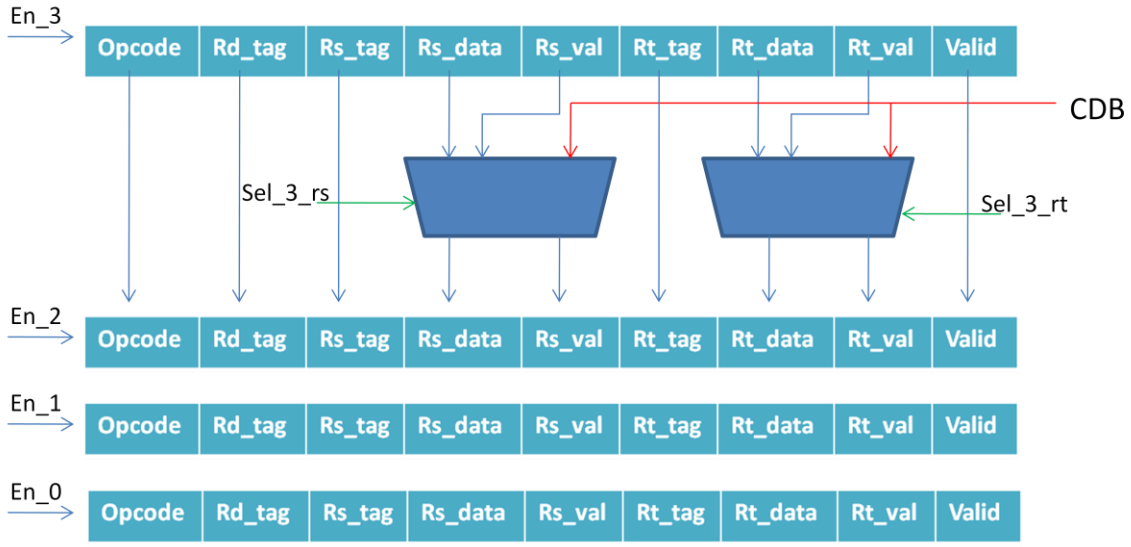


Figura 2.

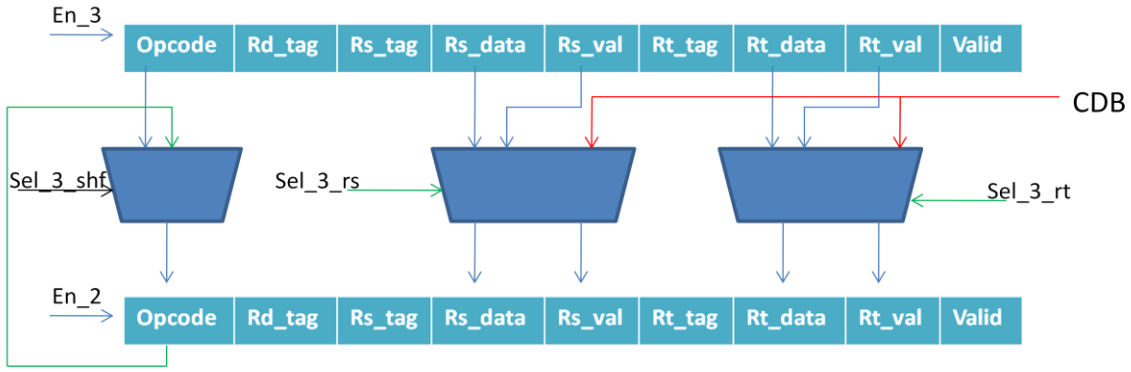


Figura 3.

Ahora bien, para poder genera las señales de control tanto para los multiplexeros como para los enables es necesario determinar si alguna entrada de la cola esta lista para ser ejecutada, comparar los datos del CDB con todas las entradas de la cola y generar la bandera de “llena”.

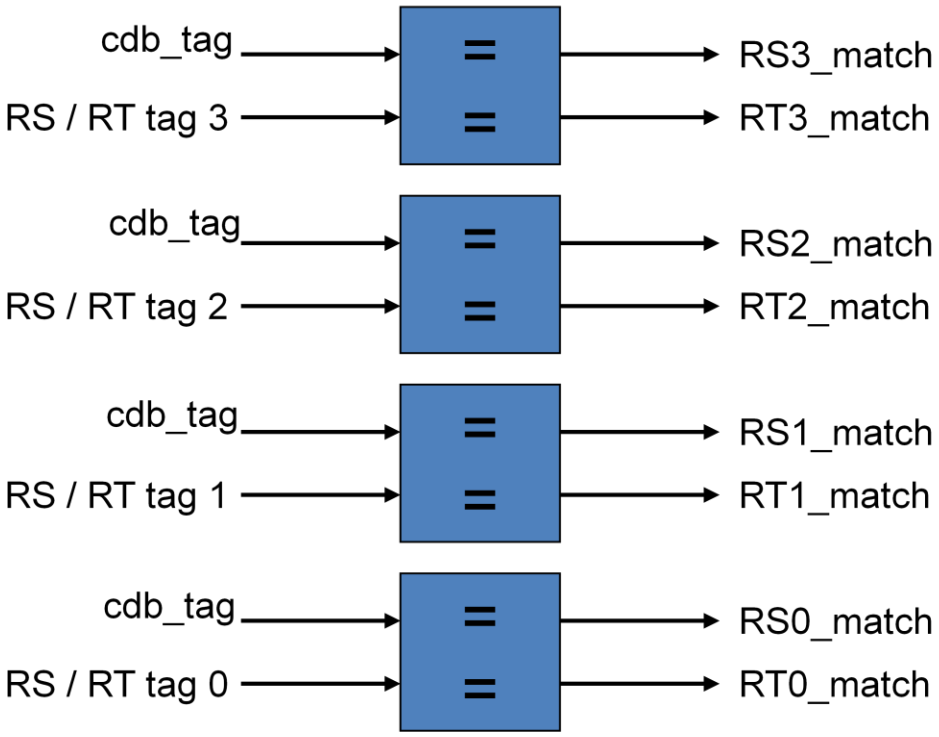


Figura 4.

La Figura 4 muestra la lógica necesaria para generar las señales de Match para cada tag fuente. La Figura 5 muestra la lógica necesaria para generar la señal issuequeue_ready. Finalmente la Figura 6 muestra la lógica necesaria para generar la bandera "llena".

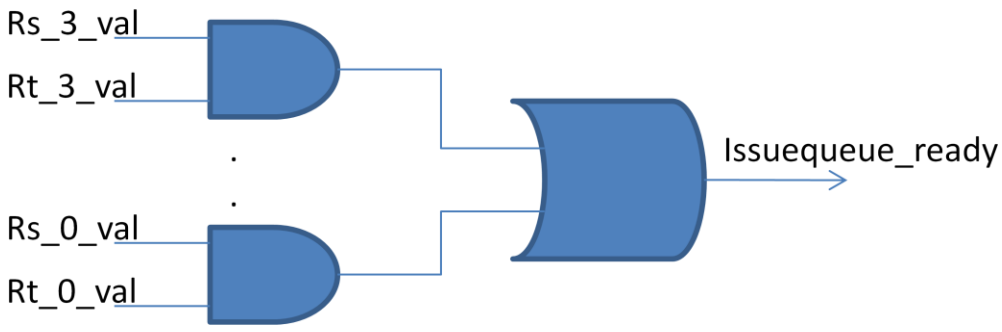


Figura 5.

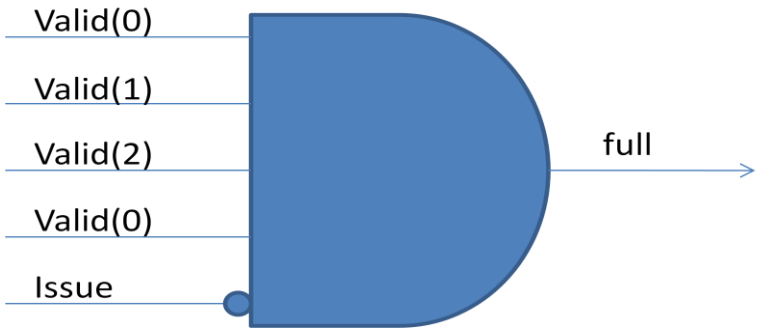


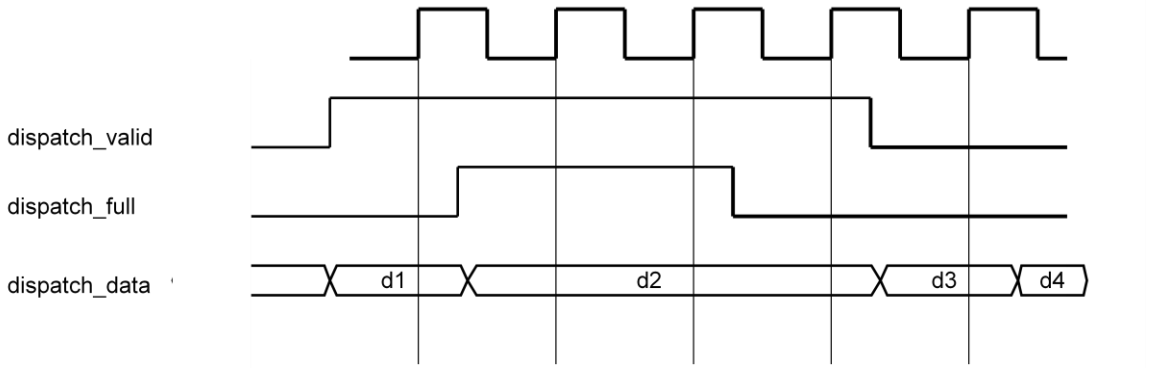
Figure 6.

Para generar la bandera “llena” es necesario tomar en cuenta no solo los campos “valid” de cada entrada de la cola sino también la señal done para así no perder un reloj cuando la cola se encuentra llena y al mismo tiempo que se desocupa una entrada se quiere ingresar una nueva instrucción a la cola.

2.2 Handshake con la unidad de despacho y con la unidad de issue.

Handshake con la unidad de despacho.

La unidad de despacho asume que una instrucción es consumida cada ciclo al menos que la señal dispatch_full se encuentra activada ‘1’.

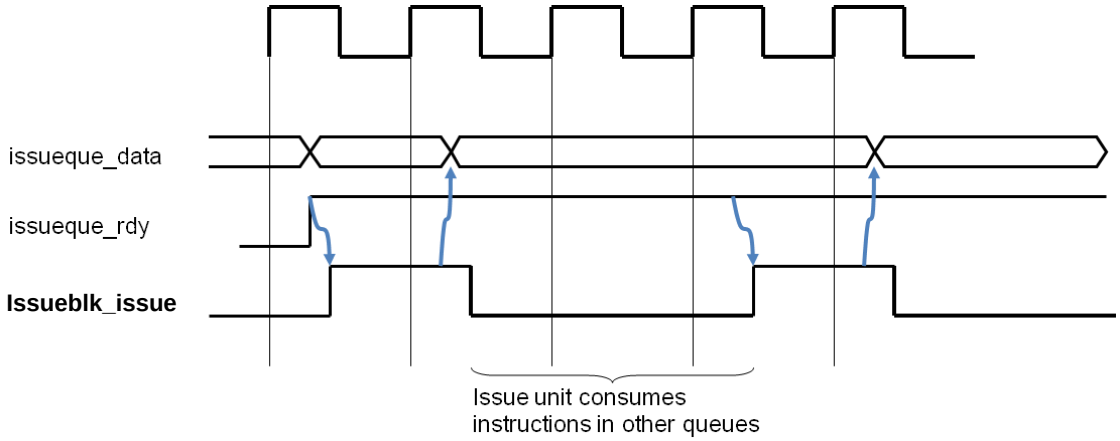


Handshake con la unidad de Issue.

1. Le indica a la unidad de Issue que tiene una instrucción lista para ejecutar activando issuequeue_ready.

2. Si la unidad de Issue decide consumir la instrucción activara combinacionalmente la señal issueblk_issue durante el mismo ciclo de reloj.

3. En el siguiente flanco de subida, si la señal issueblk_issue fue activada, se realiza la operación shift en la cola y si existe otra instrucción lista para ser ejecutada la señal issueque_ready se mantiene activa .



2.3 Salidas a la unidad de Ejecucion / Issue.

La unidad de ejecución únicamente necesita recibir los operandos y el opcode la operación que debe de ejecutar. La unidad de Issue recibe la señal de ready antes mencionada así como el tag del registro destino.

Es importante mencionar que no existe un mecanismo de “bypass” para instrucciones que entran a la cola de ejecución ya con sus operandos resueltos, siempre primero son almacenados en la cola y un ciclo más tarde son candidatos a ser ejecutadas.

En caso que existan más de una instrucción lista para ser ejecutada se tiene que seleccionar la instrucción que se encuentre mas “abajo”. Entonces para generar la señal de select de la figura 7 es necesario tomar en cuenta la posición de la instrucción en la cola.

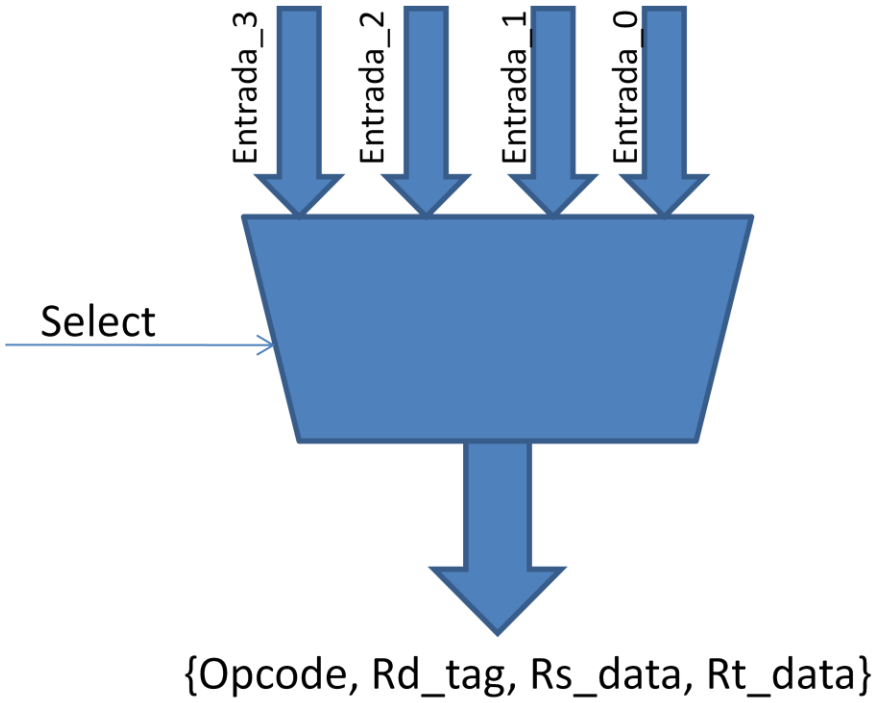

Entrada_3
Entrada_2
Entrada_1
Entrada_0
Select
{Opcode, Rd_tag, Rs_data, Rt_data}

Figura 7.