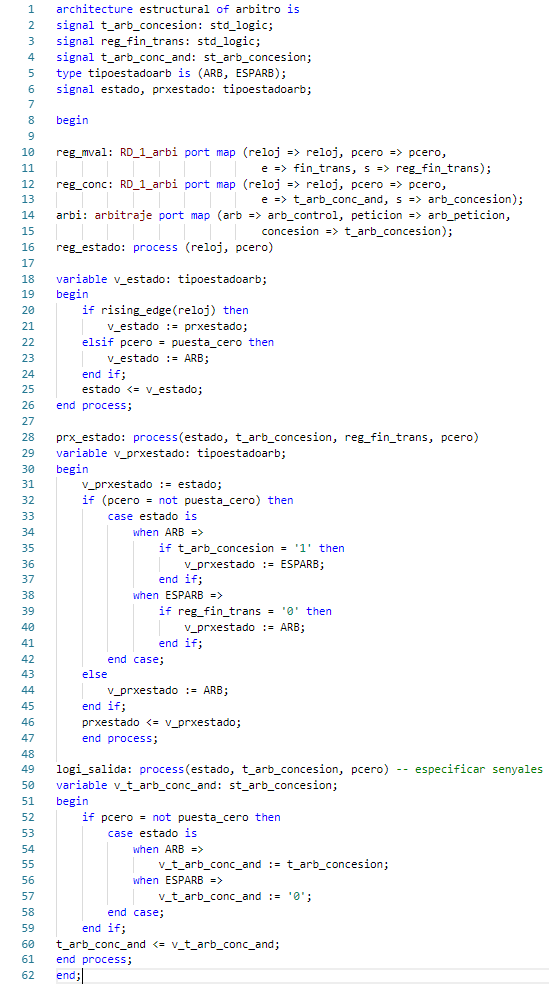
# Multiprocesadores : Práctica 3

Héctor Romero de Blas, Miquel García de Mendoza, Ramon Mateo

Trabajo 1: Codifique de forma funcional en VHDL, utilizando el constructor “process”, el autómata utilizado en el árbitro (Figura 11), identificando de forma explícita los estados., en la página 6.

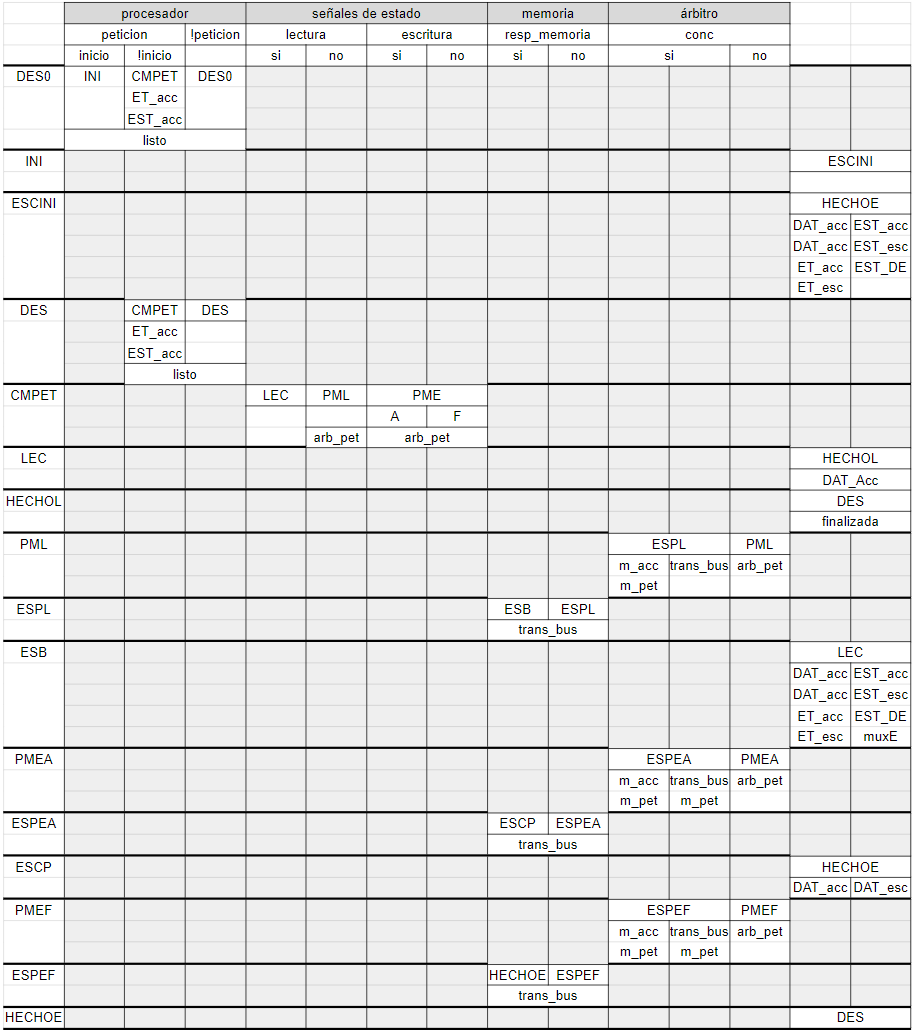


*Figura 1: descripción en VHDL del autómata para el árbitro.*

Trabajo 2: Describa la función de la puerta “and” de la Figura 15. Relaciónelo con la Figura 8., en la página 7.

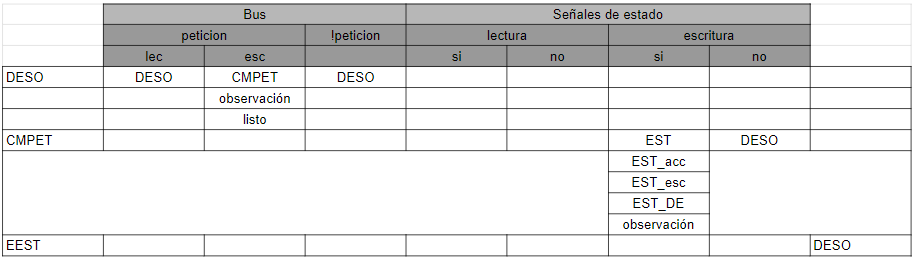
La puerta “and” sirve para guardar el valor del bus en el registro cuando es una transacción PtE, que es la que puede modificar el estado en el autómata de observación.

Trabajo 10: Construya una tabla de transiciones entre estados, donde se especifique también la lógica de salida, teniendo en cuenta el incremento de funcionalidades del agente procesador respecto del caso uniprocesador (Figura 46 en el Apéndice 3.2). La petición de acceso al bus se efectúa, si es el caso, en el estado CMPET y la concesión se espera en el siguiente estado, el cual es función del tipo de petición (Figura 19). Además, establezca el valor de la señal de ocupación del bus por parte del CC en función del estado (trans\_bus en la Figura 48 del Apéndice 3.2). En la Figura 49 del Apéndice 3.2 se relacionan los acrónimos que deben utilizarse., en la página 16.



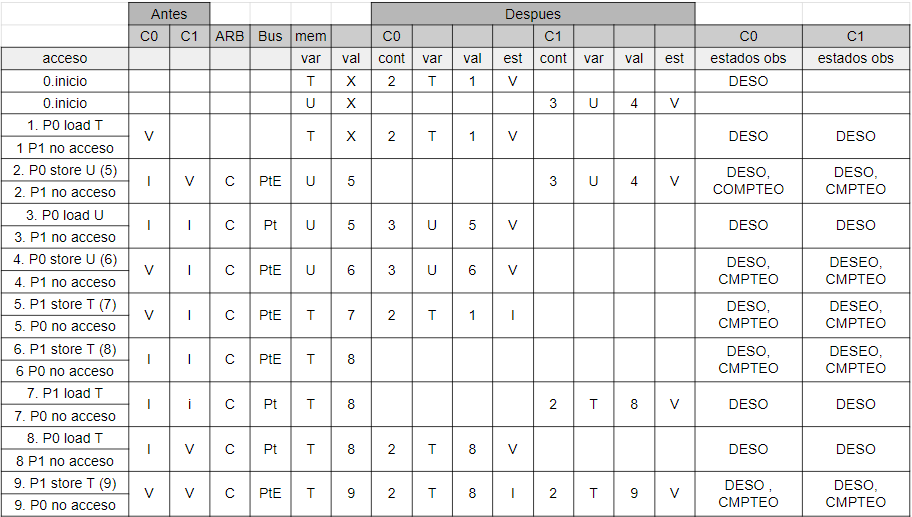
*Tabla 1: transiciones entre estados con lógica de salida especificada.*

Trabajo 12: Construya la tabla de transiciones entre estados del agente observador (Figura 47 del Apéndice 3.2), donde también se especifica la lógica de salida en la segunda subfila de cada estado (activación, desactivación de las señales). En el caso de la lógica de salida indique sólo la activación de la señal (valor 1). En la Figura 51 se relacionan los acrónimos que deben utilizarse., en la página 17.



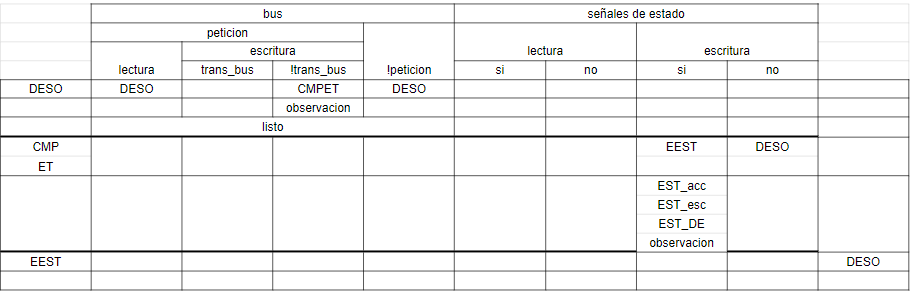
*Tabla 2: transiciones entre estados del agente observador con lógica de salida especificada.*

Trabajo 28: Construya una secuencia de accesos que muestre una comprobación incremental del diseño (directorio PRUEBAS\_obs). Razone de forma sucinta la construcción de la secuencia y las comprobaciones que se efectúan. Represente la secuencia utilizando una tabla como en la Figura 31 y añada una columna para el razonamiento., en la página 22.



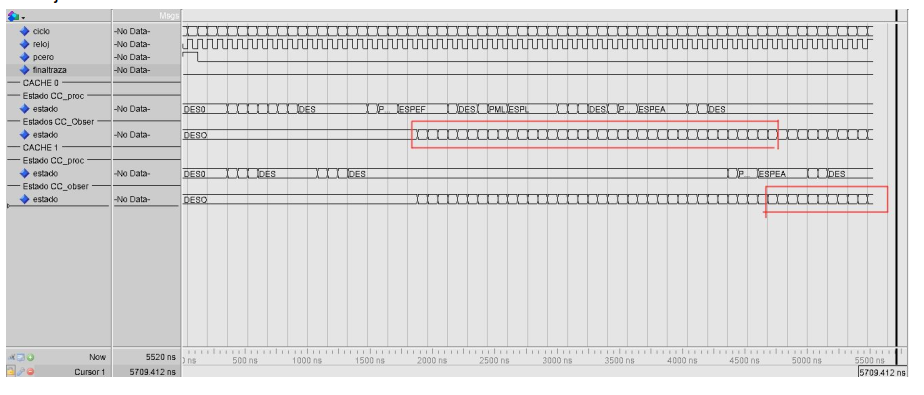
*Tabla 3: secuencia de accesos a memoria en comprobación incremental del diseño.*

Trabajo 31: Construya una tabla de transiciones entre estados del agente observador (Figura 68 en el Apéndice 3.6), donde se especifique también la lógica de salida, teniendo en cuenta el incremento de funcionalidades del agente observador (Figura 32)., en la página 23.

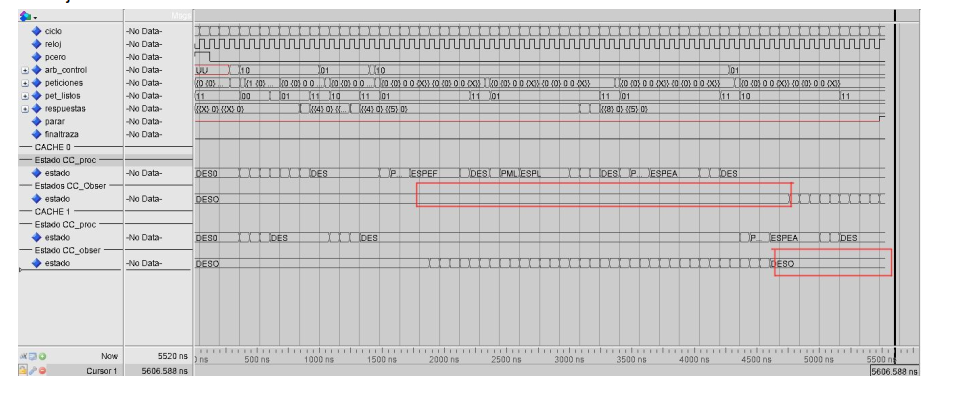


*Tabla 4: transiciones entre estados del agente observador con las nuevas funcionalidades.*

Trabajo 37: Compare los resultados del “Trabajo 34:” con los resultados del “Trabajo 25:”. Indique las diferencias en las ventanas temporales de toda la simulación, almacenadas en los ficheros, del “Trabajo 35:” y del “Trabajo 26:”, respectivamente., en la página 24.



*Figura 2: resultados de la simulación del trabajo 25.*

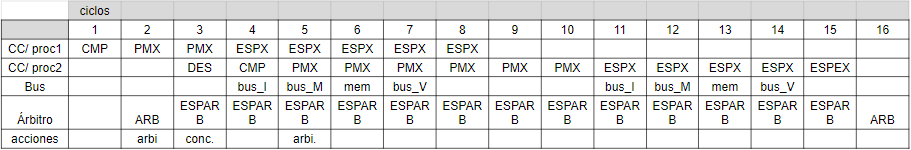


*Figura 3: resultados de la simulación del trabajo 35.*

Trabajo 41: Razone la utilización de la señal “trans\_bus” en lugar de la señal “arb\_conc” en el agente observador, con el objetivo de que no se active la operación de observación cuando un CC está utilizando el bus., en la página 24.

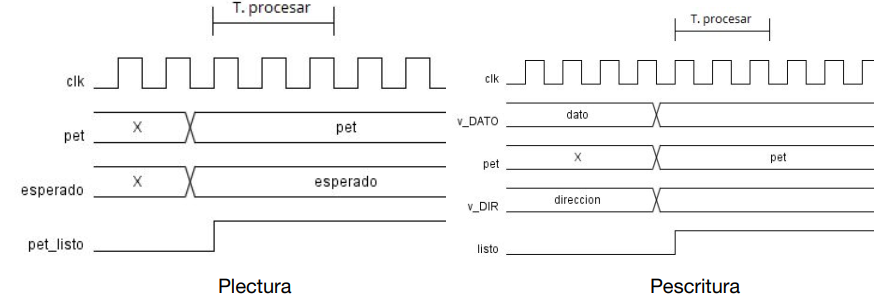
No se utiliza la señal arb\_conc porque no se mantiene activa durante toda la transacción del bus, ya que esta solo dura un ciclo.

Trabajo 42: Mientras hay una transacción ocupando el bus (1º ciclo en el estado ESPX) otro procesador efectúa una petición. El árbitro le concede el bus, a esté último procesador, cuando finaliza la transacción en curso. Muestre el encadenamiento de la ocupación del bus por parte de ambas transacciones. Para ello utilice un diagrama temporal, donde se utilice el reloj como señal de referencia. En este diagrama debe mostrar las acciones de arbitraje y los estados en los CC., en la página 24.



*Tabla 5: diagrama temporal de estados en CC y acciones arbitraje tomadas.*

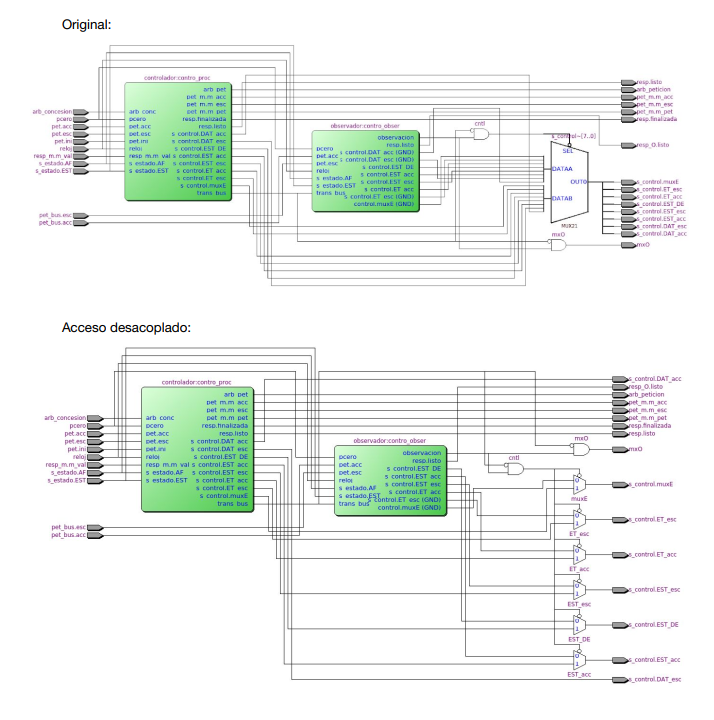
Trabajo 44: Analice el programa de prueba. Describa los procesos “productor” y “consumidor”, correspondientes a una cache, mediante diagramas temporales. Así mismo, describa los procedimientos “Plectura” y “Pescritura”. Para ello, utilice la señal reloj y sus flancos como referente., en la página 27.



*Figuras 4.1 y 4.2: diagramas temporales de los procedimientos Plectura y Pescritura.*

El proceso productor genera dos peticiones de escritura seguida de tres peticiones de escritura, dichas modificaciones se escriben en un fichero en el cual el consumidor hace lecturas y comprueba los valores con los teóricos.

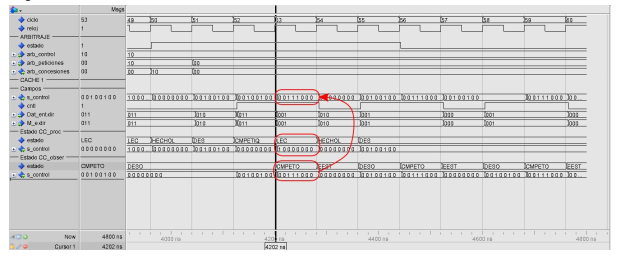
Trabajo 53: Imprima en sendos ficheros pdf, y almacénelos en el directorio RESULTADOS asociado, los diagramas RTL que genera Quartus, después de elaborar el diseño, correspondientes al camino de datos y al controlador. Identifique las diferencias entre este diseño, donde ha sido desacoplado el acceso, y el diseño previo (Trabajo 43:)., en la página 29.



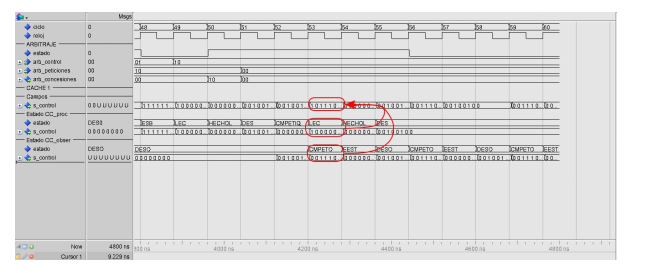
*Figuras 5.1 y 5.2: diagramas RTL del diseño original (trabajo 43) y actual con acceso desacoplado.*

En el diseño con el acceso desacoplado podemos observar cómo, en lugar de pasar todas las señales de s\_control por un solo multiplexor, cada una pasa por un multiplexor, excepto DAT\_acc y DAT\_esc, que son obtenidas directamente del autómata del procesador (en la figura 5.2 son la primera y última señal, a la derecha).

Trabajo 56: Utilizando el resultado de la simulación, describa en los ficheros, correspondientes a la ventana temporal, almacenados en RESULTADOS\_elim (Trabajo 55:) los hechos que se producen cuando se elimina el riesgo (de forma similar a la Figura 64 del Apéndice 3.4 o la Figura 78 del Apéndice 3.7)., en la página 30.



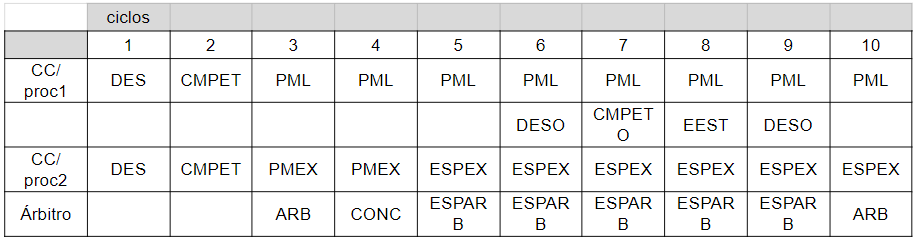
*Figura 6: resultados de la simulación con diseño original.*



*Figura 7: resultados de la simulación con diseño desacoplado.*

Al desacoplar el acceso vemos como la señal que sale del módulo controlador incluye el bit de acceso a datos que proporciona el autómata procesador, a diferencia de la ventana temporal original, donde todas las señales de control provienen del autómata observador.

Trabajo 57: Razone de forma sucinta si es factible que el contenido de un contenedor de cache, que almacena un bloque que se expulsa, debido a un fallo en una instrucción load, sea invalidado por una observación. Utilice un diagrama temporal como el de la Figura 33., en la página 30.



*Tabla 6: diagrama temporal de fallo en instrucción load y expulsión de bloque.*

El proc1 provoca un fallo de lectura, así que se queda esperando en el estado PML a que el árbitro le conceda el bus para hacer la petición a memoria. Al mismo tiempo, si se solapa la escritura en el bus de otro procesador de esta dirección con el PML, proc1 invalidará el bloque que va a expulsar.

Trabajo 74: Utilizando el resultado de la simulación, describa en el fichero, correspondiente a la ventana temporal, almacenado en RESULTADOS\_cruce (Trabajo 73:) los hechos que se producen cuando se observa el cruce., en la página 36.

No se llegó a ejecutar por errores en quartus que no llegamos a solucionar.

Trabajo 75: Razone de forma sucinta si es necesario realizar alguna actuación adicional para un funcionamiento correcto del diseño realizado en la práctica., en la página 36.

No, ya que con el diseño actual se garantiza la coherencia de las caches, dado que cada controlador puede observar adecuadamente las escrituras de los otros y en consecuencia actualizar el estado del contenedor asociado. Al usarse un bus como red de interconexión, se garantiza que todos los controladores observan el mismo orden de accesos a memoria, manteniéndose también la consistencia.

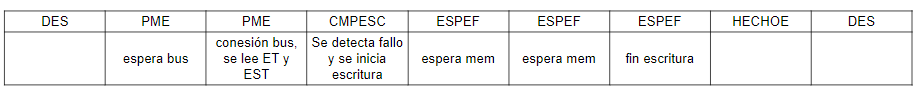
Trabajo 76: Extienda el razonamiento, sobre coherencia de cache del Trabajo 75:, a un diseño donde se utilice un buffer de escrituras y además intrucciones load, más jóvenes, que fallan en cache, puedan adelantar (acceder al bus) a instrucciones store almacenadas en el buffer de stores, si la dirección no es coincidente. En el caso de coincidencia de direcciones la instrucción load se espera hasta que desaparece la coincidencia, al irse vaciando el buffer. Además, razone de forma suscinta si este nuevo diseño mantiene consistencia secuencial., en la página 36.

En este caso, el funcionamiento también sería correcto, y además más rápido. Al producirse un store, este, aunque no se escriba inmediatamente en memoria, invalidará todas las cache que contengan el bloque (Excepto la de origen, que estará actualizada). Si alguna de estas intenta acceder al bloque, se producirá un fallo, y se deberá esperar al vaciado del buffer, conservándose la consistencia secuencial para ese bloque. Si una cache quisiese acceder a otro bloque, podría hacerlo sin problemas, tanto en lectura como en escritura. Sin embargo, la consistencia secuencial para el conjunto de todos los bloques no se conserva.

Trabajo 77: Proponga, para una petición store, una modificación del diagrama temporal (Figura 21 de la Práctica 2 y Figura 28 de esta práctica) donde sólo se visite una vez los campos ET y EST y este hecho se efectúe lo antes posible. Además, el campo DAT sólo debe actualizarse si el bloque es válido en el instante de actualización. Por otro lado, el sistema multiprocesador debe respetar consistencia secuencial. Justifique de forma precisa y concisa la respuesta., en la página 36.

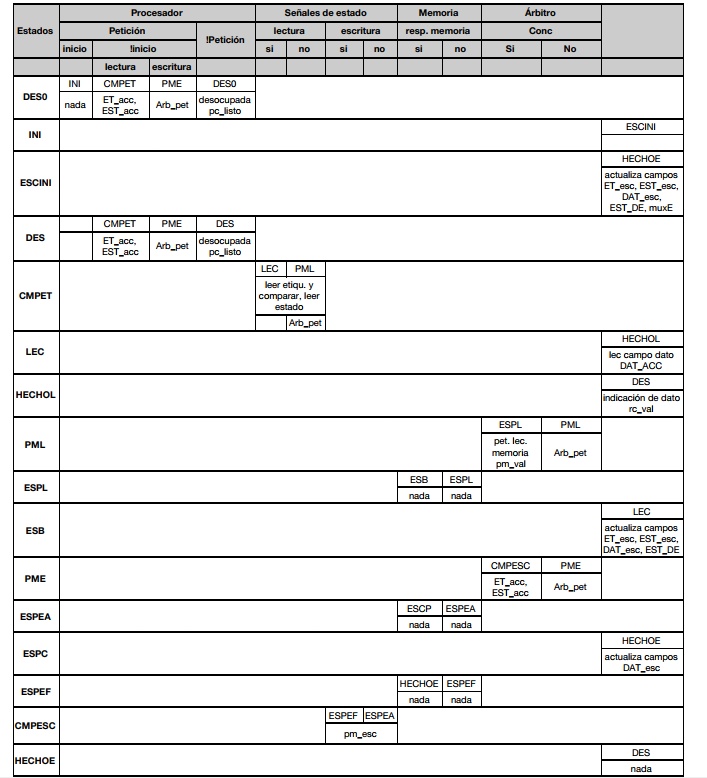
Para poder leer solo una vez los campos ET y EST, estos deberán ser leídos tras la concesión del bus. En el caso de una lectura esto no es posible, ya que implicaría pedir el bus incluso para los aciertos en caché.

Por lo tanto, se podría modificar el estado DES/DES0 para que solo realice la lectura de los campos en caso de lectura. En caso de escritura, realizaría la petición de arbitraje y pasaría a un nuevo estado (PME) hasta la concesión de bus. Al concederse el estado, se leerían los campos ET y EST pasando a un nuevo estado (CMPESC). En este nuevo estado se pasaría a PMEF o PMEA en función de lo leído en el estado anterior, eliminando la actualización innecesaria de DAT.



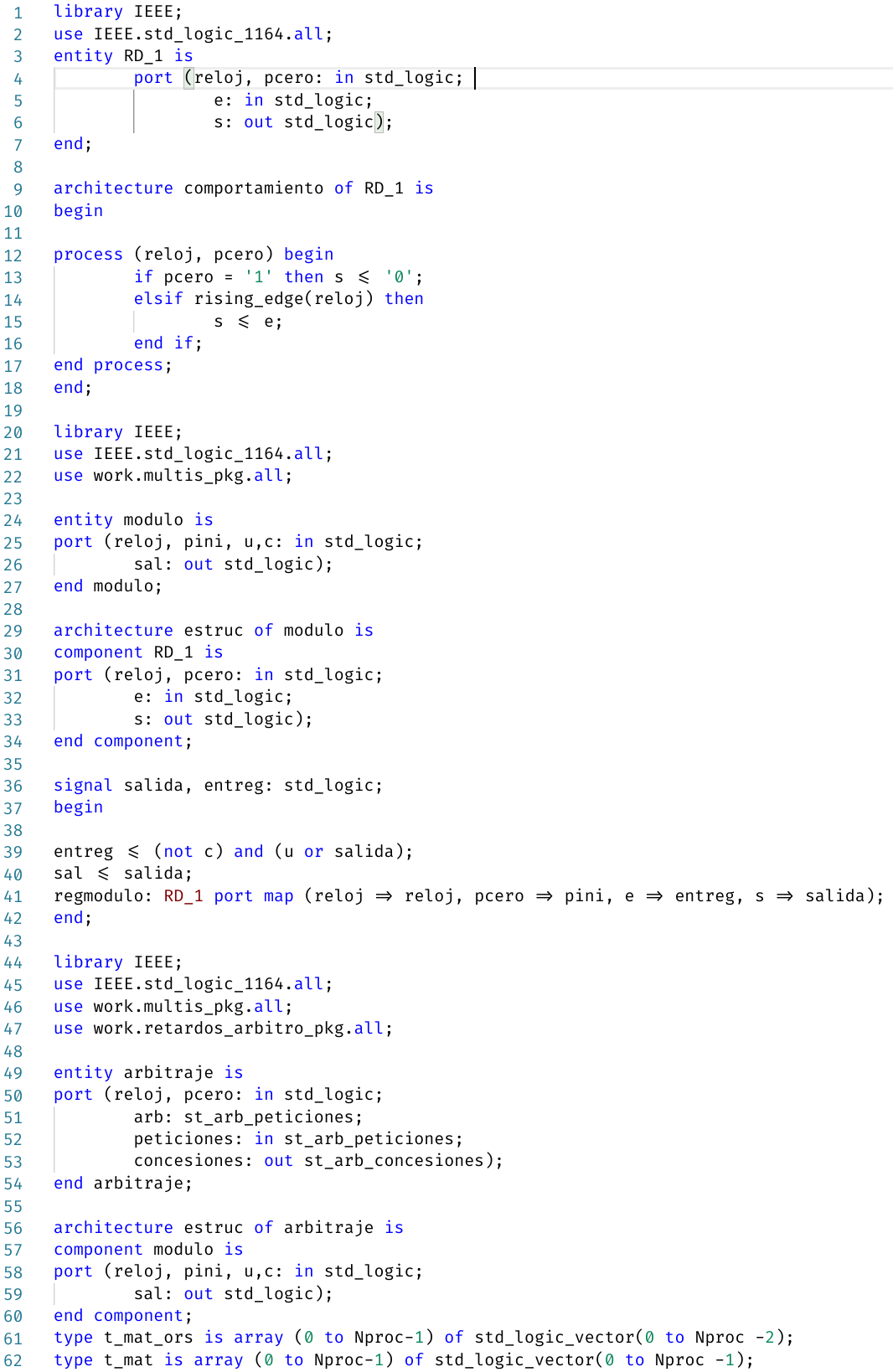
*Tabla 7: propuesta de modificación del diagrama temporal.*

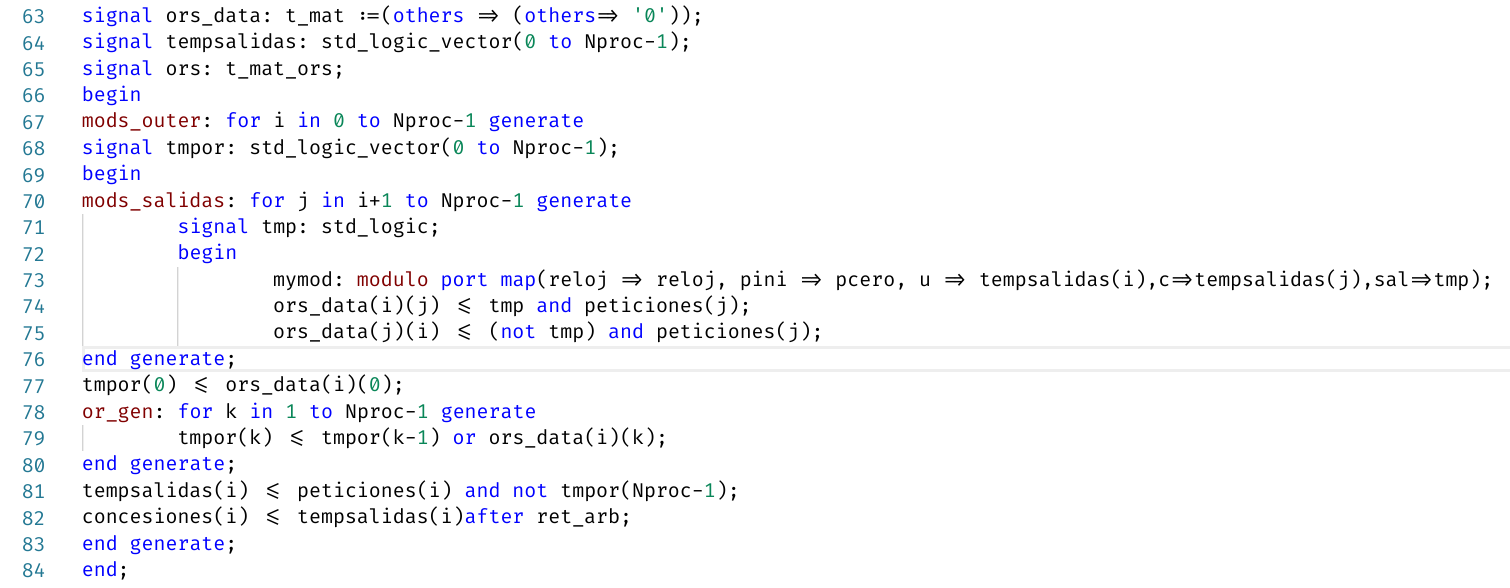
Trabajo 80: Construya una tabla de transiciones entre estados, donde se especifique también la lógica de salida, que se corresponda con el autómata del Trabajo 79:., en la página 37.



*Tabla 8: transiciones entre estados con especificación de la lógica de salida.*

Trabajo 84: Describa en VHDL de forma paramétrica (para cualquier número de entradas), mediante un diseño estructural, un árbitro matricial donde sólo se utiliza la parte superior de la matriz de arbitraje (Figura 109). Para ello, en primer lugar, debe de especificar el elemento que se utiliza para almacenar el estado y la lógica asociada para establecer valores (Figura 106)., en la página 38.

*Figura 8: código VHDL del árbitro* 



*Figura 9: código VHDL del árbitro (continuación)*