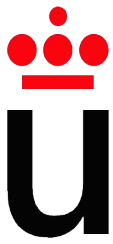


Universidad
Rey Juan Carlos

Arquitectura e Ingeniería de Computadores

Práctica 1: Diseño de la unidad de control de un procesador MIPS multiciclo

David Rozas Domingo (47456048-X)
4º curso de Ingeniería Informática Superior

Metodología de implementación

Para la realización de esta práctica se ha seguido un modelo de desarrollo espiral, implementando incialmente las operaciones tipo R y la operación LOAD, y comprobando su correcto funcionamiento.
Posteriormente se agregaron el resto de operaciones obligatorias : STORE, BEQ y JUMP. Además se ha implementando también la operación BNE.

Se ha seguido el diagrama de estados del guión original, teniendo que agregar dos estados más (9 y 10) para la fase de ejecución de las operaciones JUMP y BNE.

Los ficheros creados/modificados que se adjuntan son:

UC.vhd: Implementación de la unidad de control. Para poder seguir de forma más sencilla los resultados de los tests, se ha agregado una señal de tipo natural que lleva el nº de ciclos totales de ejecución.

TEST.vhd: Aplica los estímulos necesarios al procesador.

MIPS.vhd: Entidad que conecta la unidad de control y el datapath.

memoria_ins.vhd: Entidad que modela la memoria de instrucciones. Se han cambiado para realizar una multiplicación.

Todos ellos se encuentran en el directorio *source*. Se adjuntan además los pantallazos de las pruebas por si la resolución en este documento no fuera la suficiente. Todas ellas se encuentran en el directorio *screenshots*.

Test efectuados para comprobar el funcionamiento: código ensamblador de una multiplicación

Para comprobar el correcto funcionamiento de las instrucciones, se ha creado un código sencillo en el que primero se prueba la instrucción JUMP, y posteriormente se realiza una multiplicación y se guarda su resultado en memoria.

El código es el siguiente:

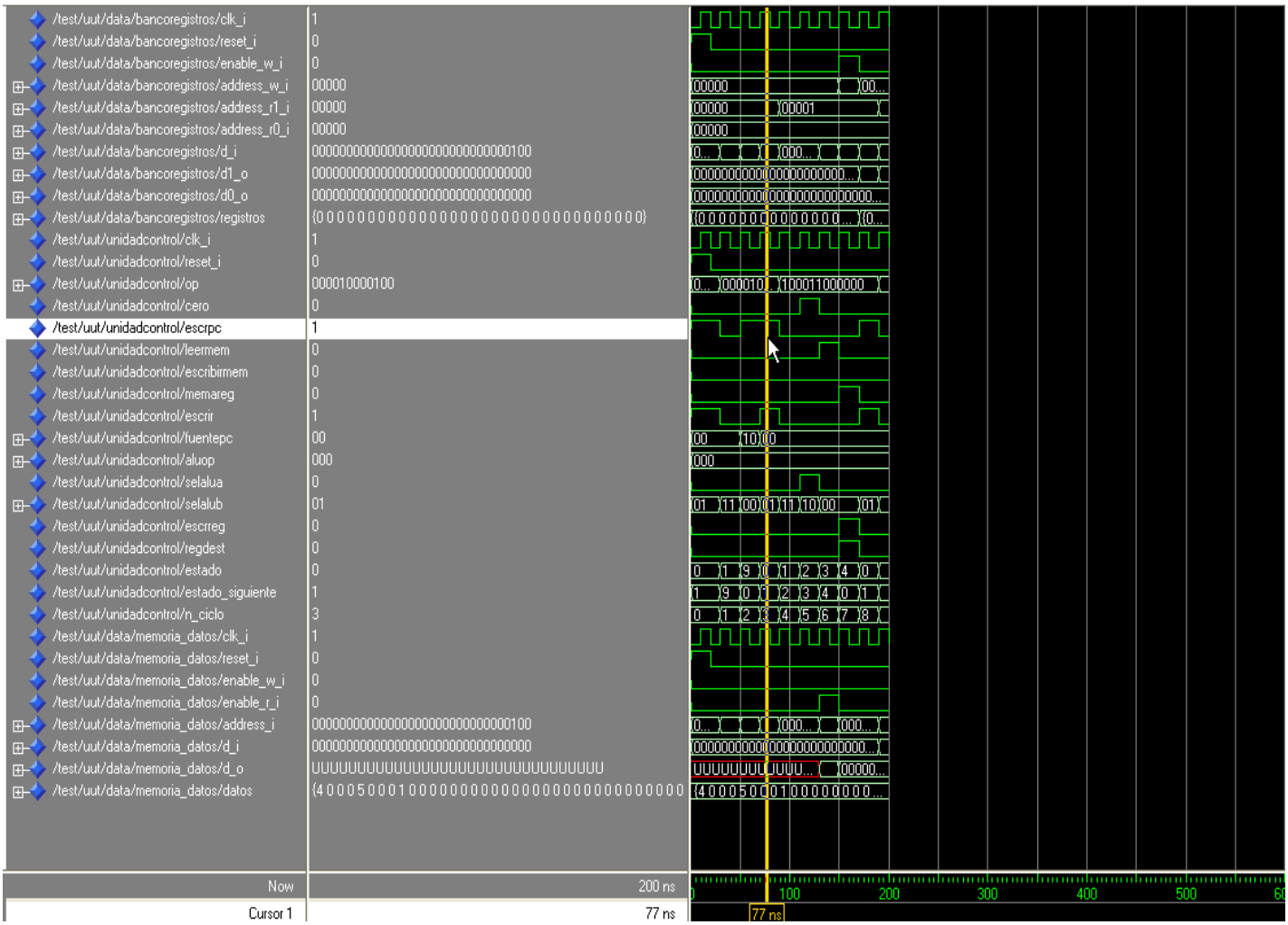
```
0  jump 4
4  lw  r1,0(r0)
8  lw  r2,4(r0)
12 lw  r3,8(r0)
16 lw  r1,0(r0)
20 lw  r2,4(r0)
24 lw  r3,8(r0)
28 add r5,r5,r1
32 sub r2,r2,r3
36 bne r2,r0,-3
40 sw r5,12(r0)
```

La descripción del flujo de ejecución de dicho código se detalla en el apartado siguiente.

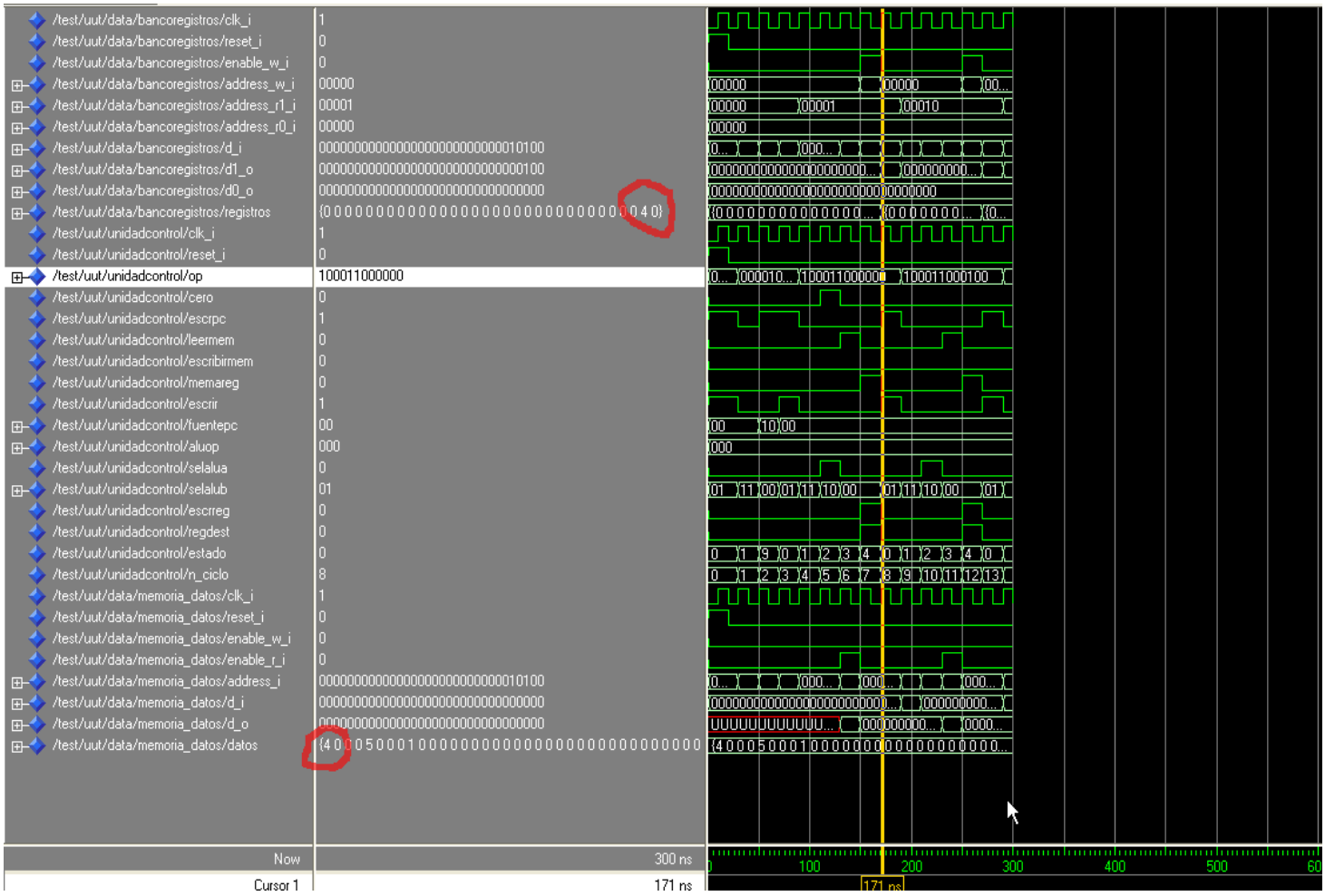
Descripción del flujo de ejecución

A continuación, se describe el proceso de ejecución de las instrucciones en los ciclos más importantes.

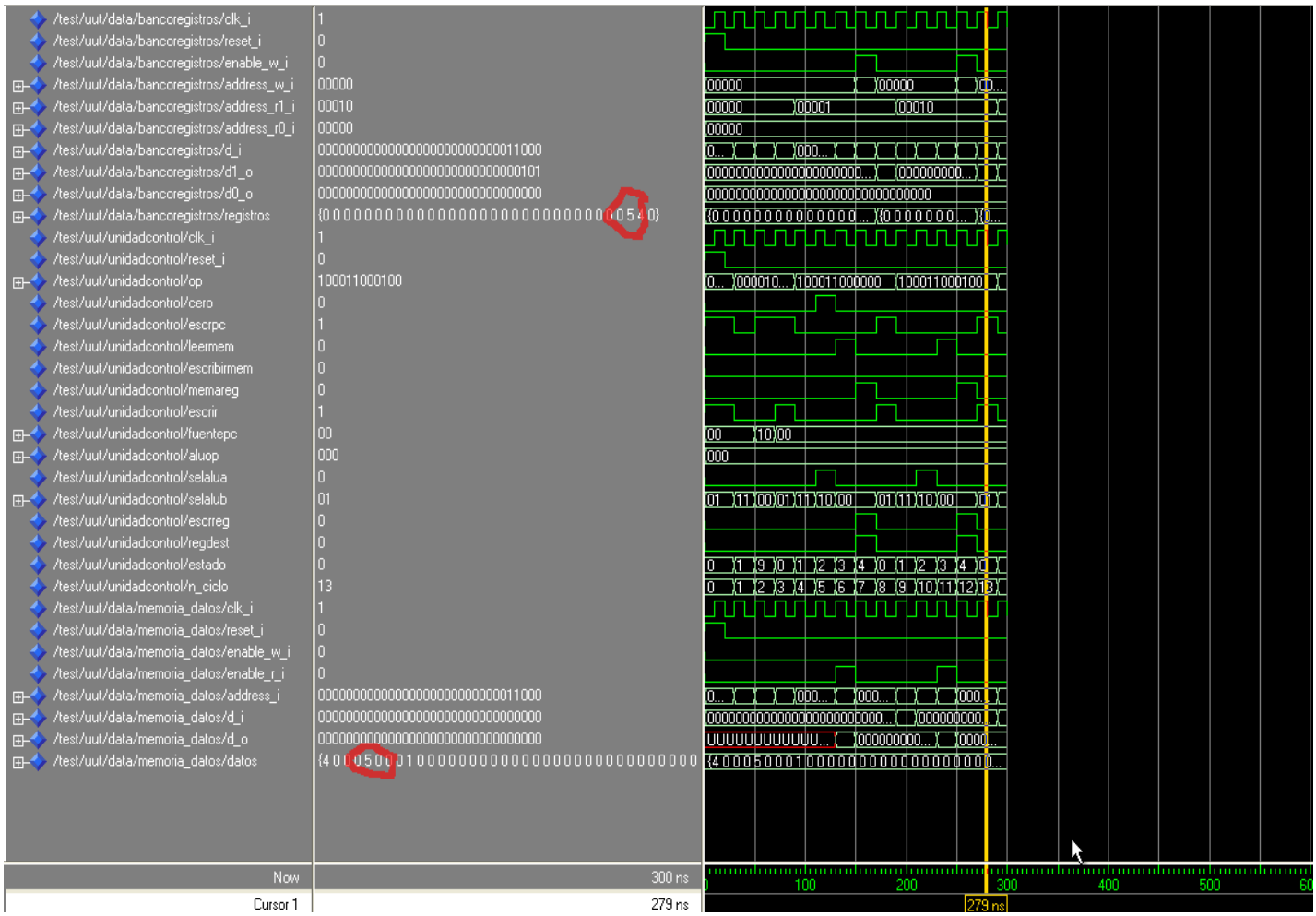
 # Ciclo 3: Se ha ejecutado la instrucción JUMP 4 durante los ciclos 0..2 pasando por los estados 0, 1 y 9. Pasaremos a ejecutar las quinta instrucción.



 # Ciclo 8: Se ha ejecutado la instrucción LW R1, 0(R0) duante los ciclos 3..7, pasando por los estados 0, 1, 2, 3 y 4.
Dicha instrucción ha cargado en el registro R1 el contenido de la memoria de datos de la posición (R0=0 + desp=0) = $0, que es el número 4.

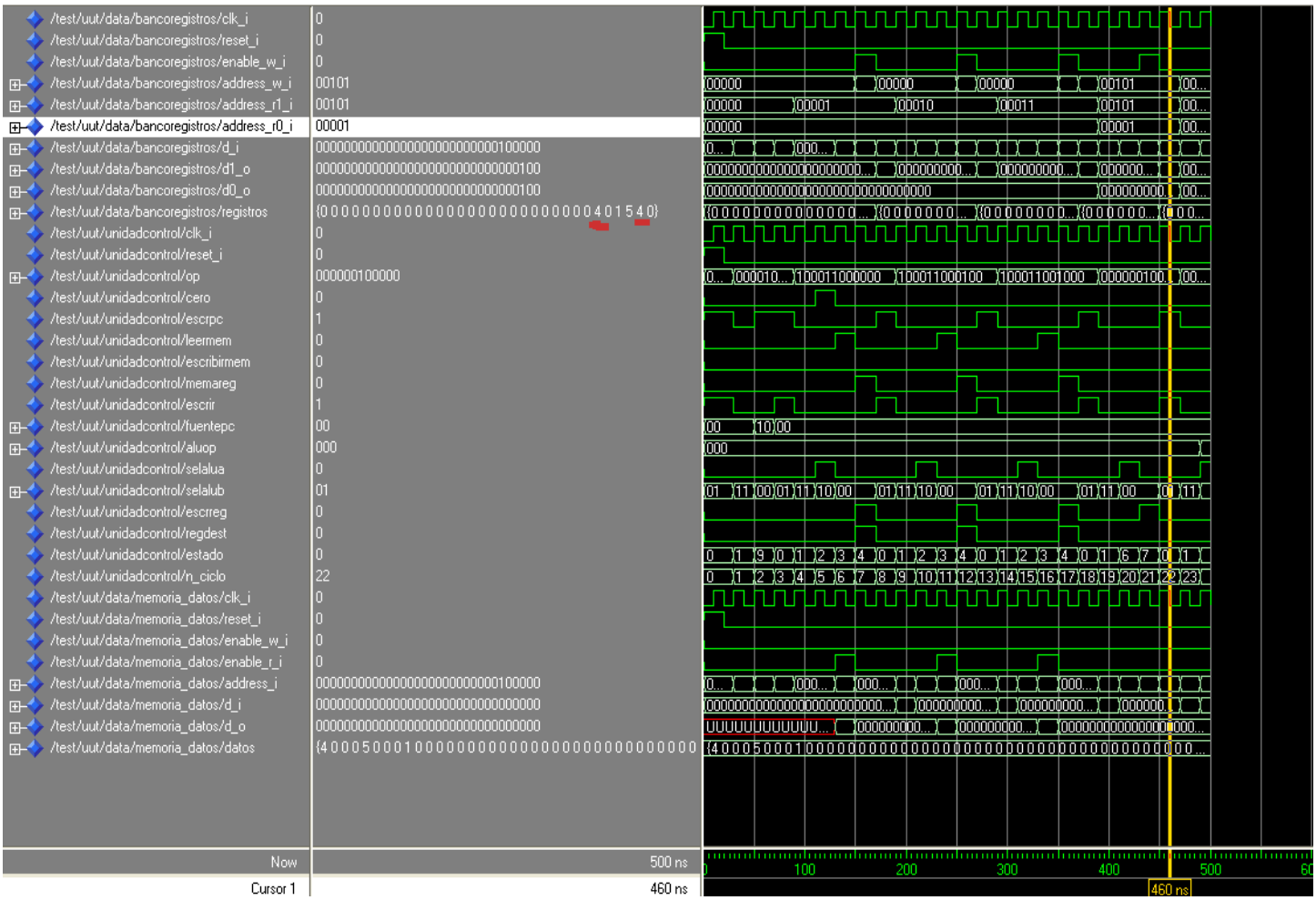


 # Ciclo 13: Se ha ejecutado la instrucción LW R2, 4(R0) durante los ciclos 8..12, pasando por los estados 0, 1, 2, 3 y 4.
Dicha instrucción ha cargado en el registro R2 el contenido de la memoria de datos en la posición (R0=0 + desp=4) =$4, que es el número 5.

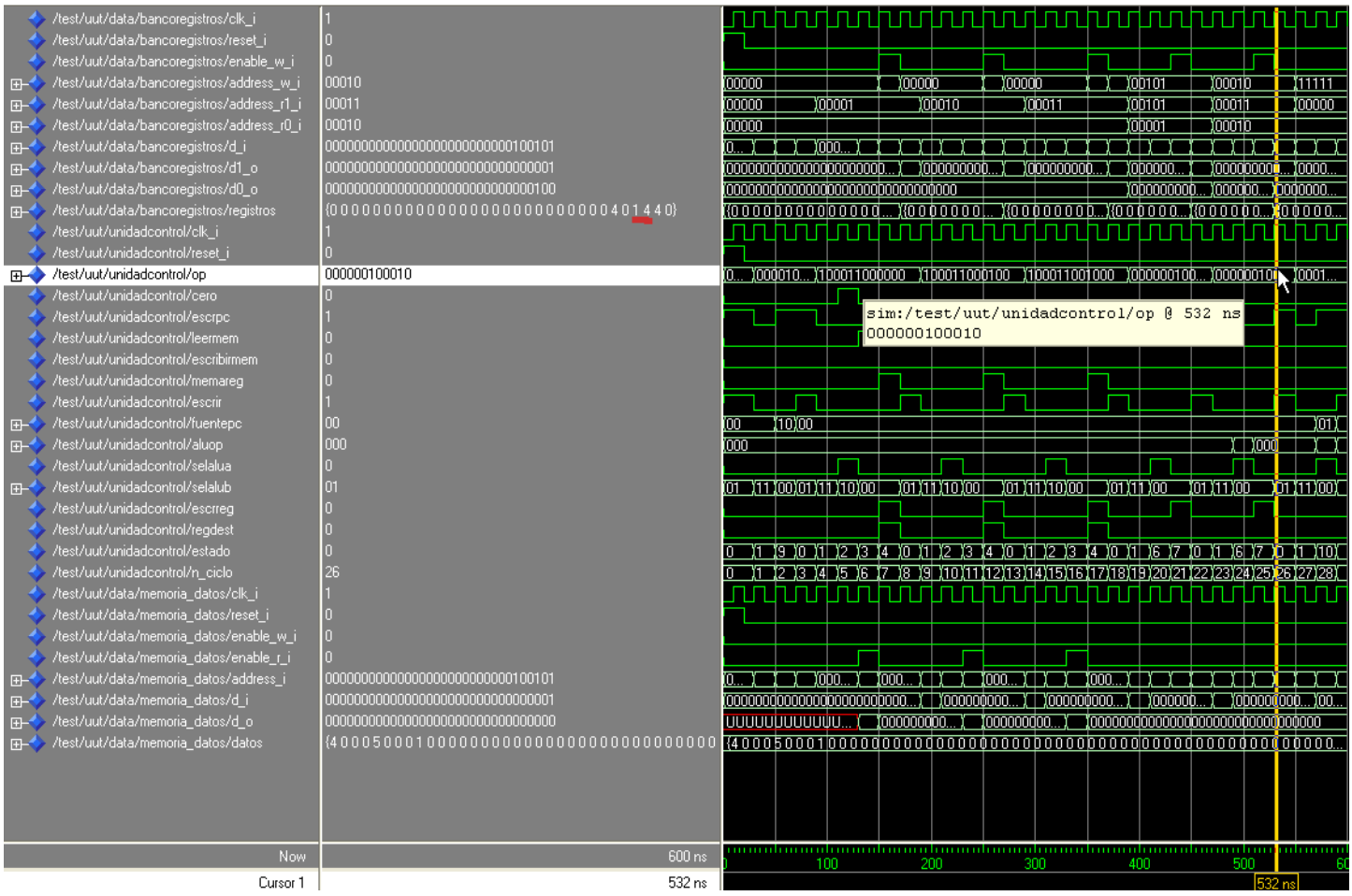


 # Ciclo 18: Se ha ejecutado la instrucción LW R3, 8(R0) durante los ciclos 13..17, pasando por los estados 0, 1, 2, 3 y 4.
Dicha instrucción ha cargado en el registro R3 el contenido de la memoria de datos en la posición (R0=0 + desp=8) = $8, que es el número 1.

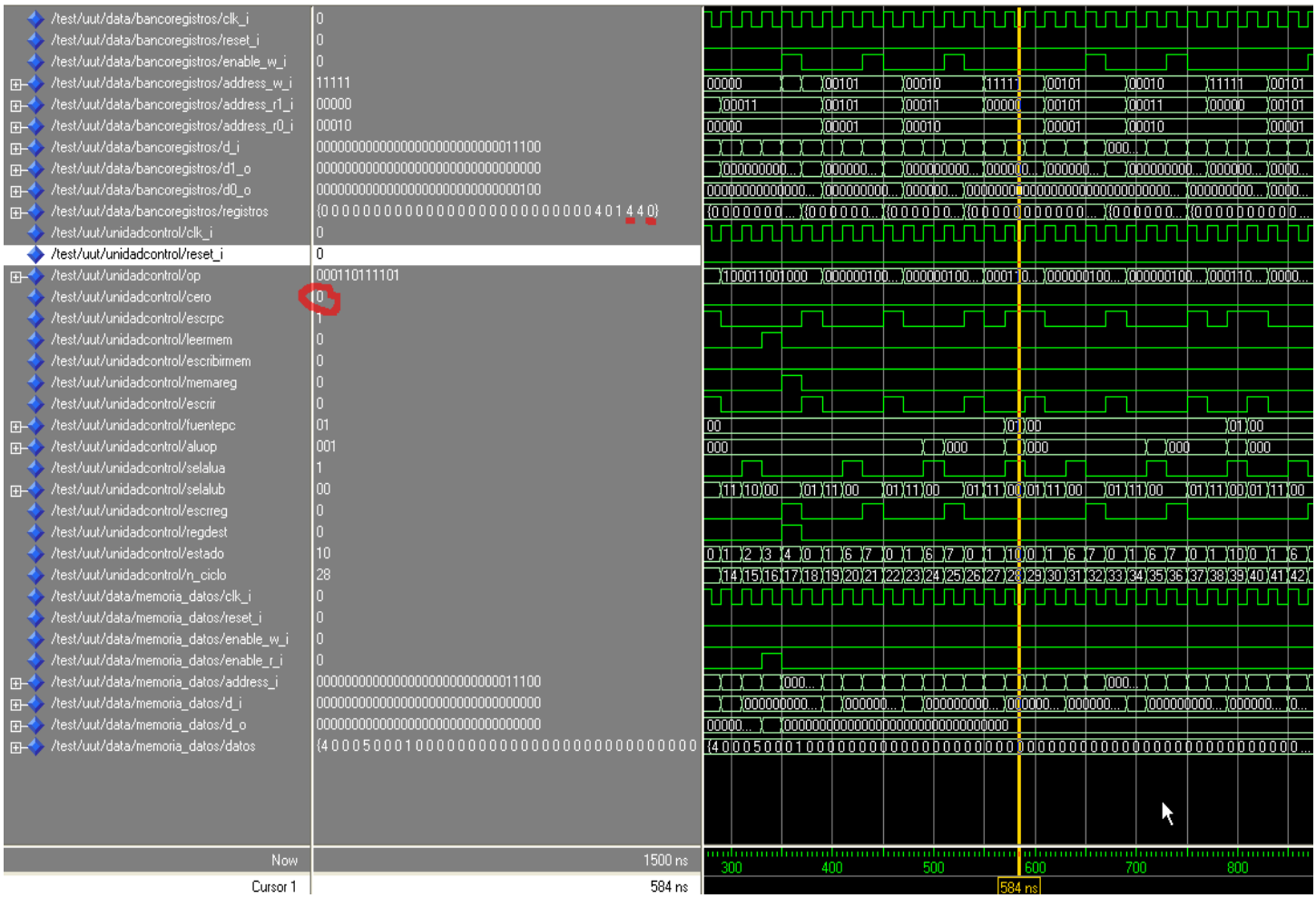
 # Ciclo 22: Se ha ejecutado la instrucción ADD R5, R5, R1 durante los ciclos 18..21, pasando por los estados 0, 1, 6 y 7.
Dicha instrucción ha sumado en el registro R5 los valores de R5=0 y R=4. Por tanto, R5=4. En dicho registro, es en el que se almacenará el resultado de la mutltiplicación.



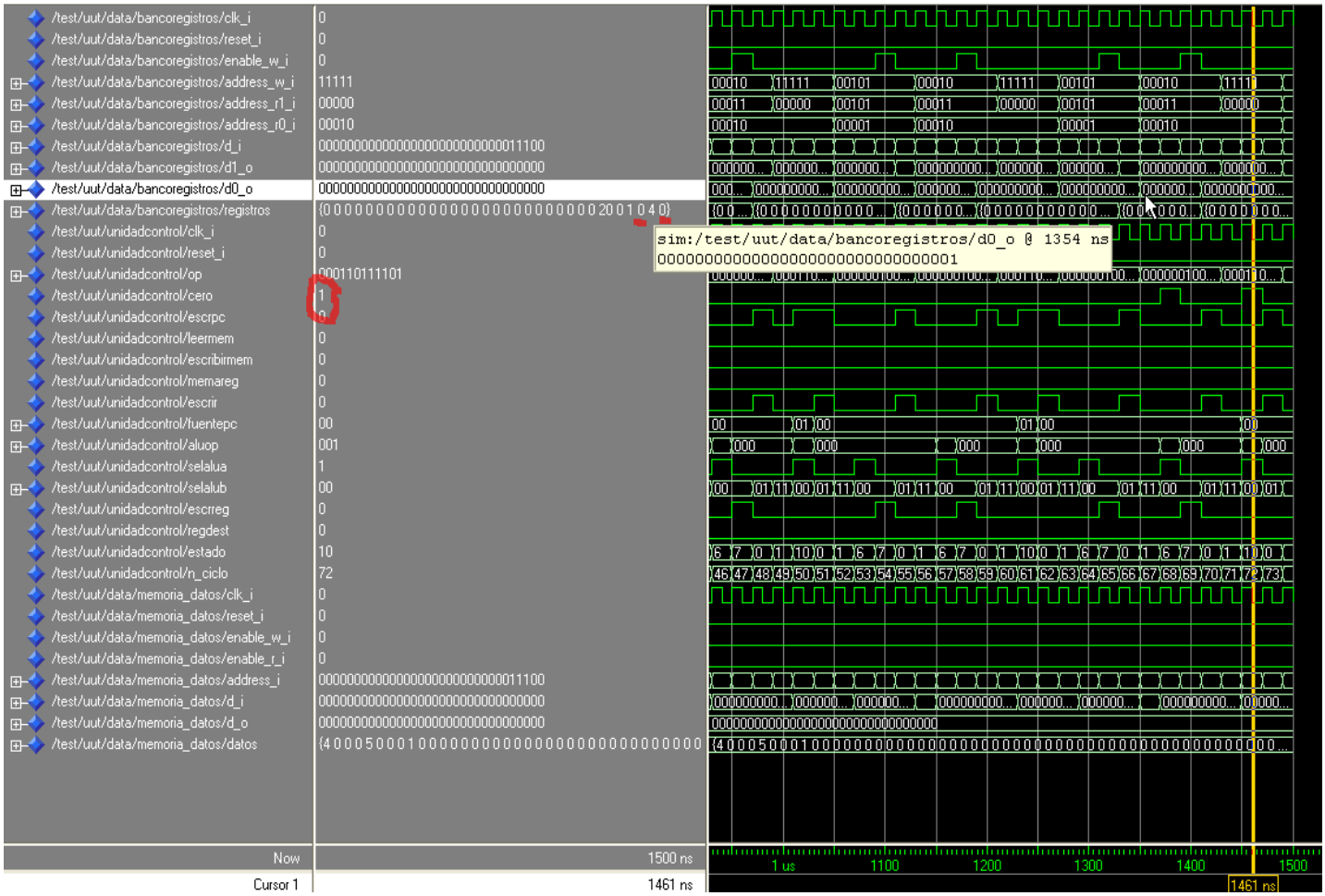
 # Ciclo 26: Se ha ejecutado la instrucción SUB R2, R2, R3 durante los ciclos 22..25, pasando por los estados 0, 1, 6 y 7.
Dicha instrucción ha restado R2=5 y R1=1, y guarda su valor en R2, que pasa ahora a ser 4. Dicho registro es el que contador de nuestro bucle, que iremos decrementando en una unidad.



 # Ciclo 28: Se va a ejecutar la instrucción BNE R2, R0, -3. Puesto que R2 = 4 y R0 = 0, la señal *cero* seguirá siendo 0, y saltaremos atrás con un desplazamiento de -3.
Es decir se van a ejecutar las operaciones de suma y resta durante 4 iteraciones más, hasta que R2 sea 0.



 # Ciclo 72: Se va a ejecutar la instrucción BNE R2, R0, -3. En este momento R5 es igual a 20, y R2 ya ha sido decrementado hasta 0. Por tanto, ahora al comparar R0 y R2 el valor de cero es 1, y no procedemos a realizar el salto hacia atrás.



 # Ciclo 77: Se ha ejecutado la instrucción SW R5, 12(R0) durante los ciclos 73..76, pasando por los estados 0, 1, 2 y 5.
Dicha instrucción guarda en la posición de memoria (R0=0 + desp=12) = $12 el valor de R5, que es en el que hemos almacenado el resultado de nuestra multiplicación 4 x 5 = 20.