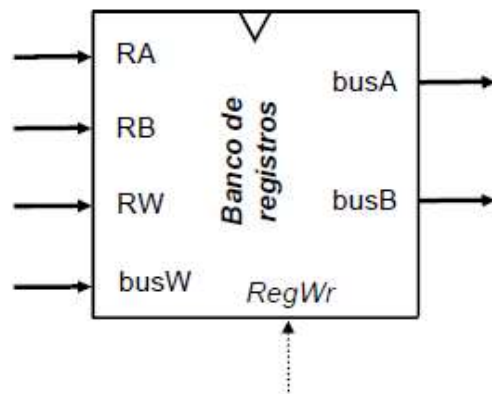


DISEÑO DE UN BANCO DE REGISTROS PARA UN PROCESADOR MIPS

Práctica 1 (1ª sesión)



Arquitectura y Organización de Computadores 2
2º Grado Ingeniería Informática

Luis M. Ramos
luisma@unizar.es

Alejandro Valero
alvabre@unizar.es

José Luis Briz
briz@unizar.es

Javier Resano
jresano@unizar.es



Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza



Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

1 RESUMEN

En esta práctica vamos a diseñar un banco de registros para un procesador MIPS. Seguiremos un procedimiento de diseño modular, utilizando bloques combinacionales: registros, multiplexores y decodificadores. Una vez diseñado, comprobaremos su correcto funcionamiento y realizaremos un análisis temporal.

Al entrar al laboratorio coloca el trabajo previo del apartado 2.1 encima de la mesa, de forma visible. Antes de empezar a trabajar con Logisim conéctate a Moodle desde uno de los equipos del laboratorio y realiza el contrato de prácticas y el control de asistencia a la sesión.



La práctica finaliza cuando el banco de registros funciona correctamente y han sido entregados a través de **Moodle** el **circuito del apartado 2.2.h** y los **resultados del apartado 3**.





2 DISEÑO DE UN BANCO DE REGISTROS 32x32

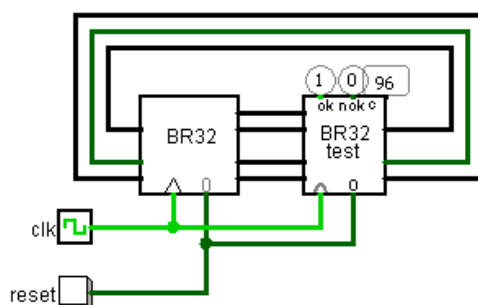
2.1 TRABAJO PREVIO: PRIMEROS PASOS CON LOGISIM Y DISEÑO MODULAR

- Lee el recurso [“Primeros pasos con Logisim”](#) y ve el video [“Logisim Beginner's Tutorial”](#), disponibles en Moodle.
- Diseña sobre papel un banco de 4 registros de 32 bits (**BR4**), utilizando cuatro registros de 32 bits, dos multiplexores MUX 4:1x32 y un decodificador DECOD 2:4. El banco de registros debe permitir dos lecturas y una escritura en cada ciclo. En las entradas **RA** y **RB** se indica qué registros se quieren leer, obteniéndose los datos en **busA** y **busB**. Si **RegWr**=1 el dato en **busW** se escribirá en el registro **RW** en el flanco ascendente de final de ciclo.
- Diseña un banco de 32 registros de 32 bits (**BR32**) utilizando 8 **BR4**, dos MUX 8:1x32 y un DECOD 3:8. Recuerda que debe permitir dos lecturas y una escritura en cada ciclo. Utiliza los mismos nombres en entradas y salidas que en el **BR4**.

2.2 TRABAJO EN EL LABORATORIO: DISEÑO CON LOGISIM

- Coloca 4 registros (carpeta de componentes *Memoria*) en una hoja de Logisim. Configúralos como registros de 32 bits (*propiedades/bits de datos*). Coloca un DECOD 2:4 (carpeta *Plexores/Decodificador* y *propiedades/seleccionar bits* → 2) cuyas salidas controlen los permisos de escritura de los registros (**en**). Interconecta las señales de **reset**, **reloj** y de entrada de **datos**.
- Coloca dos MUX 4:1x32 (*Plexores/Multiplexor*, *propiedades/seleccionar bits* → 2, *propiedades/Bits de Datos* → 32 y *propiedades/Include Enable?* → No) que seleccionen el dato de alguno de los registros.
- Coloca entradas  para las señales **reset**, **reloj** y **busW**, **RA**, **RB**, **RW** y **RegWr**. Configúralas con el número de bits correcto (*propiedades/Bits De Datos*) y nómbralas (*propiedades/etiqueta*). Conéctalas donde corresponda. Conecta salidas  a los MUX, configúralas de 32 bits y nómbralas (**busA** y **busB**).
- Ya tenemos un banco de 4 registros. Comprueba que funciona correctamente realizando varias escrituras (**RW**, **busW**, **RegWr**), lecturas (**RA**, **RB**) y borrados (**reset**).

- e) Renombra el circuito como **BR4** (clic en  main / propiedades / Nombre Del Circuito). Puedes cambiar la apariencia del encapsulado y la posición de los pines pulsando . Pula  para volver al diseño del circuito.
- f) Añade un circuito nuevo (Proyecto / Añadir circuito). Nómbralo **BR32**. Márcalo como circuito principal (Proyecto / Seleccionar Como Circuito Principal).
- g) Coloca 8 **BR4**. Añade un DECOD 3:8 para que la escritura se efectúe correctamente en uno de los 32 registros. Añade MUX 8:1x32 para que las dos lecturas se efectúen correctamente.
- h) Coloca entradas y salidas para todas las señales del banco de registros de 32 bits (también para la señal de reloj). Configúralas del número de bits adecuado. Utiliza separadores para separar los bits necesarios (Wiring/Separador) y conectarlos donde correspondan. Utiliza túneles (Wiring/Tunnel) para conectar señales a través de nombres.
- i) Cambia a modo *simulación*  y realiza varias escrituras y lecturas en distintos registros. Para visualizar el valor de cada señal puedes añadir componentes Ver (Wiring/Ver) o directamente hacer clic en la señal. También puedes hacer doble clic en un **BR4** para entrar en él y comprobar el contenido de los registros.
- j) Antes de realizar la entrega del circuito debes pasarle un vector de pruebas. Para ello:
- Asegúrate de que no usas ningún componente *Reloj* (la señal de reloj debe estar conectada a una entrada normal).
 - Añade un circuito nuevo. Nómbralo **main**. Márcalo como circuito principal.
 - Ve al recurso Moodle “Entrega NIA-BR32test”, descarga el fichero “BR32test.circ” y guárdalo en el mismo directorio que tu fichero .circ. Añádelo como componente externo (Proyecto / Cargar Librería / Librería Logisim / Seleccionar fichero “BR32test.circ”).
 - En el circuito **main** coloca un componente **BR32test** y tu componente **BR32**. Coloca un botón de *reset* y un componente *Reloj* y conéctalos a las entradas de ambos componentes.
 - Conecta las salidas de **BR32test** (**RegWr**, **busW**, **RW**, **RA**, **RB**) a las entradas de **BR32**. Conecta las salidas de **BR32** (**busA**, **busB**) a las entradas de **BR32test**.
 - Coloca 3 componentes *Ver* en las salidas **ok**, **nok**, y **c** de **BR32test**. Configura el componente *Ver* de la señal **c** como “Unsigned decimal”.
 - Activa el reloj (Ctrl-K). Si al cabo de 96 ciclos se activa la señal **ok** tu circuito funciona correctamente. Si en algún momento se activa la señal **nok** y **c** se para tu circuito tiene algún fallo.



¿Qué hace el vector de pruebas?

ciclo (c)	
0-31	$r_i=1$ sin RegWr
1-32	$(r_i == 0)$ en RA y RB
33-64	$r_i=i+1$
34-64	$(r_o == 1)$ en RA y RB
65-96	$(r_i == i+1)$ en RA y RB

- k) Cuando tu circuito haya pasado el vector de pruebas puedes entregarlo a través del recurso Moodle “Entrega NIA-BR32test”.

3 ANÁLISIS TEMPORAL

- a) Realiza un análisis temporal de los bloques combinacionales (MUX 4:1, MUX 8:1, DECOD 2:4, DECOD 3:8), calculando el retardo de los caminos entre todas las entradas y salidas de cada bloque. Asume una implementación siguiendo la expresión $Z = X_0 * \overline{S_1} * \overline{S_0} + X_1 * \overline{S_1} * S_0 + \dots$ para los MUX y $Z_0 = en * \overline{S_1} * \overline{S_0}$; $Z_1 = en * \overline{S_1} * S_0$; ... para los DECOD. Los retardos de las puertas utilizadas son: $d_{NOT} = 5$ ps; $d_{OR2-4} = 20$ ps; $d_{AND2-4} = 20$ ps; $d_{REG} = 50$ ps; $t_{setup_{REG}} = 30$ ps. Ten en cuenta que disponemos de puertas OR de un máximo de 4 entradas.
- b) Identifica los caminos combinacionales que intervienen en la lectura del **BR32** ($RA \rightarrow busA$ y $REG \rightarrow busA$) y calcula sus retardos ($d_{RA \rightarrow busA}$, d_{busA}). Ten en cuenta que los bits de RA tienen distintos caminos, y deberás obtener el retardo máximo de todos ellos.
- c) Identifica los caminos combinacionales que intervienen en la escritura del **BR32** ($RW \rightarrow REG$, $busW \rightarrow REG$ y $RegWr \rightarrow REG$). ¿Con qué antelación al flanco de reloj tiene que establecerse el valor de cada entrada ($t_{setup_{RW}}$, $t_{setup_{busW}}$, $t_{setup_{RegWr}}$)?
- d) Cuando tengas todos los retardos y los tiempos de *setup* introdúcelos a través del recurso “Análisis temporal BR32” de Moodle.