

## PRÁCTICA 4. Trabajo final

### Implementación de un multiplicador de números naturales con la unidad de proceso general (UPG) y una unidad de control de propósito específico.

#### Objetivos que se deben alcanzar al realizar la práctica

Después de realizar esta práctica, además de haber mejorado el nivel de consecución de los objetivos necesarios para preparar la práctica (citados en la tabla anterior) el alumno será capaz de:

Diseñar e implementar dos versiones de un multiplicador de números naturales con Entradas/Salidas síncronas formado por una unidad de control de propósito específico y la UPG. En la primera versión la multiplicación tarda siempre el mismo número de ciclos mientras que en la segunda versión (denominada de terminación temprana) el número de ciclos depende del valor concreto de los bits del multiplicador. En concreto:

- a) Dibujar el grafo de estados de la unidad de control de propósito específico del multiplicador.
- b) Dibujar el esquema lógico del circuito que implementa el grafo de estados de la unidad de control del multiplicador mediante dos memorias ROM y un registro.
- c) Implementar en LogicWorks la unidad de control del multiplicador mediante dos memorias ROM y un registro.

#### Directorio de la práctica

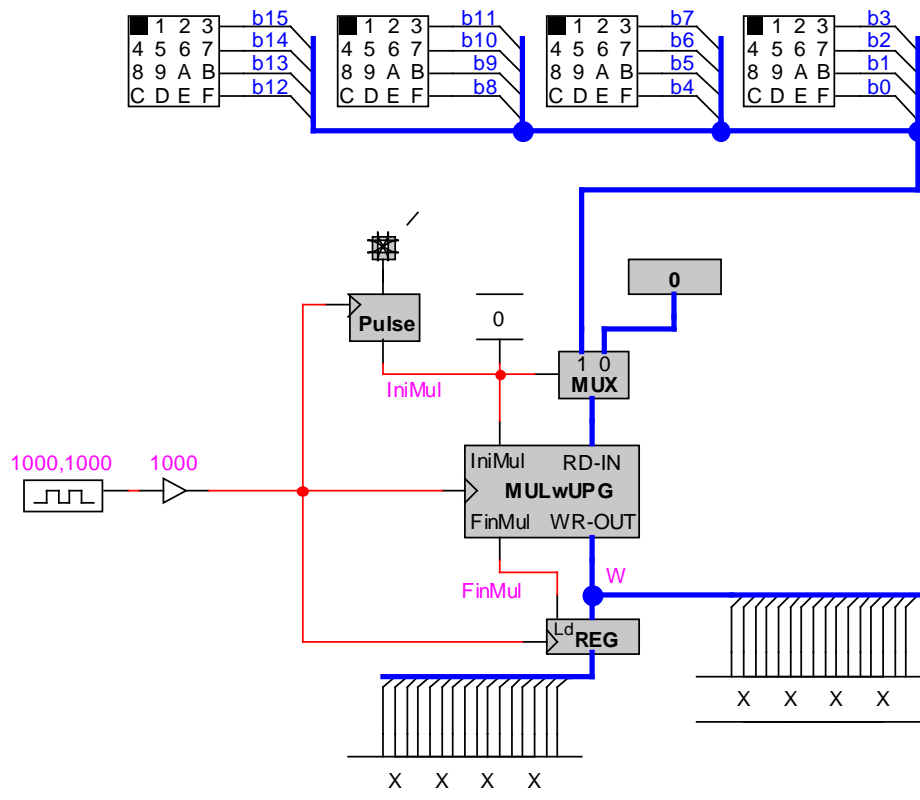
Mi PC\\:\ic\Prac4

### 1.1 Introducción

En esta práctica implementamos un multiplicador secuencial de dos números naturales codificados en binario con 16 bits. Es equivalente al diseño de propósito específico que hicimos en la práctica 3 pero ahora usando la unidad de proceso general, UPG, que hemos estudiado en clase. Como la UPG ya está diseñada, nos centramos en el diseño de la unidad de control (UC). Ahora, en el laboratorio, vamos a implementar el algoritmo con terminación temprana cuyo grafo de la UC propusisteis en la pregunta 8 del informe previo. La comunicación de este multiplicador con el subsistema que le proporciona los datos y recoge el resultado es síncrona. Recordamos ahora el protocolo de E/S:

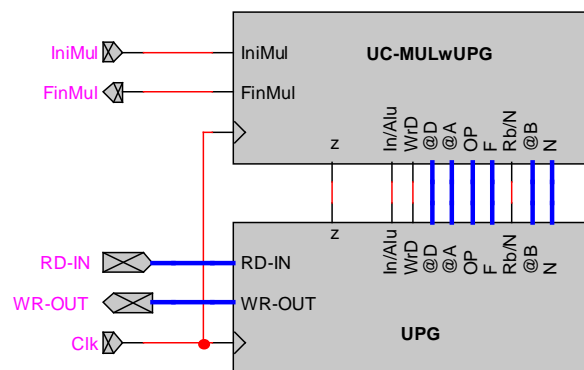
- En el primer ciclo que IniMul vale 1, el primer operando está disponible en el bus RD-IN. Pasados varios ciclos desde el primer pulso de IniMul (el número de ciclos depende de lo rápido o lento que sea el usuario en pulsar el teclado) llegará otro pulso para indicar que el segundo operando está disponible en el bus RD-IN. Sólo se puede asegurar que un operando está un ciclo en la entrada, el ciclo en el que IniMul vale 1.
- Cuando el multiplicador tenga el resultado correcto en el bus de salida, lo indicará poniendo a 1 durante un ciclo la señal de control FinMul. El resultado sólo estará presente en el bus de salida del multiplicador durante ese ciclo.
- Si mientras se está multiplicando un par de números (después del segundo ciclo en el que IniMul vale 1 y antes de que FinMul valga 1) se recibe por la entrada IniMul un 1, el multiplicador hará caso omiso de esta petición y continuará sin inmutarse hasta que ponga a 1 FinMul.
- En el último ciclo, en el que FinMul vale 1, si IniMul vale 1 comenzará otra multiplicación y en dato presente en este ciclo en RD-IN, es el primer operando de una nueva multiplicación.

Utilizaremos el circuito ProbeMxx se muestra a continuación para probar y visualizar nuestro multiplicador con terminación temprana.



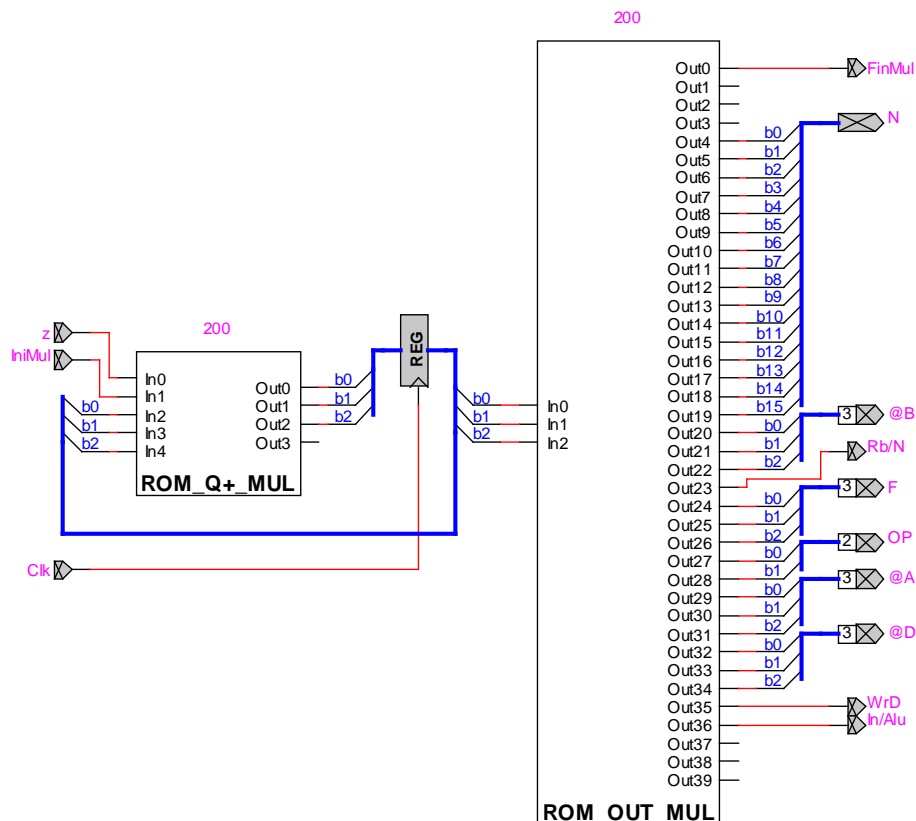
## 1.2 Implementación del multiplicador en LogicWorks


En esta sección vamos a implementar en logicWorks el multiplicador con terminación temprana usando la UPG. La siguiente figura muestra el esquema interno del multiplicador en dos bloques, la unidad de control específica y la unidad de proceso general UPG.



En la siguiente figura mostramos el esquema interno de la unidad de control específica para la multiplicación. La hemos diseñado con dos memorias ROM, la ROM\_Q+\_MUL y la ROM\_OUT\_MUL. Observad que de los 4 bits (un dígito hexadecimal) de menor peso de la salida de la ROM, solamente usamos el bit de menor peso, para codificar la señal FinMul. Hemos usado un dígito en vez de un bit para codificar FinMul, para que sea más fácil escribir el contenido de la ROM en hexadecimal, aunque por esto se necesite una ROM con 3 bits más por palabra. Además hemos usado 9 dígitos hexadecimales para la palabra de control, de 33 bits, aunque no se usen los 3 bits de más peso de la ROM.

El circuito ProbeMxx, que se encuentra en la carpeta Prac4, es una implementación completa del multiplicador que hemos visto aquí junto con los dispositivos para entrar datos y visualizar el resultado. El problema es que no funciona, ya que el contenido de las ROM de la unidad de control no es correcto.



Implementad en el laboratorio, usando el LogicWorks, cada una de las dos ROM que habéis diseñado en la pregunta 10 del informe previo. Os recordamos que construir un dispositivo de tipo ROM tenéis que seguir los pasos que os indica el *PROM/RAM/PLA Wizard*  que se encuentra en la barra de herramientas del LogicWorks. Después de hacer clic en el icono tenéis que seleccionar *PROM* (ya que la ROM que usamos es realmente una *Programmable ROM*, PROM). En la siguiente ventana poned el número de señales binarias de entrada (*Address Lines*) y el de señales de salida (*Bits per Word*), seleccionad *Enter hex data manually* y por último entrad los datos del contenido de la ROM en hexadecimal, separando cada palabra con un espacio. Podéis hacer clic en *Format Help* para saber cómo tenéis que separar cada palabra del contenido de la ROM.

Como las ROM son bastante grandes para entrar los datos manualmente es fácil cometer errores. Hay un problema, que una vez creada una ROM no se puede ver su contenido ni modificarlo. Si se quiere cambiar el contenido hay que crear otra ROM. Así si se produce un error al entrar los datos es muy difícil detectarlo y corregirlo. Podéis escribir el contenido de la ROM en un fichero tipo texto (con el bloc de notas, por ejemplo) y recortar y pegar su contenido en la ventana de entrada de datos manual.

Para que los símbolos de las ROM que creéis tengan el mismo tamaño que las que nosotros hemos puesto en el circuito de la unidad de control, tenéis que darles el mismo nombre: *ROM\_Q+\_MUL* y *ROM\_OUT\_MUL*. Si dais nombres con más caracteres, cuando tengáis que sustituir en el circuito vuestras ROM por las nuestras, no tendréis espacio suficiente y tendréis que mover componentes del circuito, cosa engorrosa. Una vez creadas vuestras dos ROM, eliminad las dos ROM que se encuentran en la unidad de control del multiplicador y colocad las vuestras. Haced reset del sistema y comprobad que el multiplicador funciona correctamente.

#### □ Informe final

##### Pregunta 1:

Cuando estéis seguros del correcto funcionamiento del multiplicador con terminación temprana, avisad al profesor de laboratorio y pedidle que revise vuestro trabajo y firme en el informe final.

#### □ Informe final

##### Pregunta 2:

Indicad si realizasteis correctamente el grafo de la pregunta 9 y el contenido de las ROM de la pregunta 10 del informe previo. En caso de que haya algo incorrecto en los grafos o en el contenido de las ROM del informe previo decid el qué es incorrecto y por qué es incorrecto. Copiad aquí el grafo correcto y el contenido de las ROM correctos.

## Informe final Práctica-4

Apellidos y nombre: ..... Grupo: .....

Apellidos y nombre: ..... Grupo: .....

(por orden alfabético)

### **Pregunta 1** (8 puntos)

Avisad al profesor de laboratorio y pedidle que revise vuestro trabajo y firme este informe final.

Comentario del profesor:

Firma del profesor:

### **Pregunta 2** (2 puntos si pregunta 1 correcta)

Indicad si realizasteis correctamente el grafo de la pregunta 9 y el contenido de las ROM de la pregunta 10 del informe previo. En caso de que haya algo incorrecto en los grafos o en el contenido de las ROM del informe previo decid el qué es incorrecto y por qué es incorrecto. Copiad aquí el grafo correcto y el contenido de las ROM correctos.

Grafo correcto de la Unidad de Control, especificando la palabra de control con mnemotécnicos.

Contenido correcto de las dos ROMs de la Unidad de Control

|            |
|------------|
| ROM_Q+_MUL |
|            |

|             |
|-------------|
| ROM_OUT_MUL |
|             |