Presentación

## Objetivos

El objetivo principal de este proyecto ha sido el diseño e implementación de un procesador tolerante a fallos transitorios. Para ellos se ha dividido en 3 tareas:

* Diseñar e implementar el procesador.
* Dotar al diseño de tolerancia a fallos transitorios.
* Y comprobar el funcionamiento del microprocesador.

# PROCESADOR

## Procesador

El microprocesador se ha diseñado para ejecutar instrucciones del repertorio ARM. En concreto ejecuta instrucciones del repertorio THUMB-2 propio de la gama CORTEX. Estas instrucciones fueron diseñadas para reducir la densidad del código, es decir, utilizar menos instrucciones para realizar el mismo trabajo que en otros repertorios. Y se codifican en 32 bits.

El procesador se ha segmentado en 5 etapas para conseguir aprovechar el HW al máximo.

Se ha utilizado la arquitectura Harvard que permite acceder a la memoria de instrucciones y a la memoria de datos en un mismo ciclo de reloj.

Además posee 16 registros de propósito general para almacenar los datos.

## Instrucciones

Las instrucciones que ejecuta el procesador se dividen en tres tipos: “Acceso a memoria”, “Procesamiento de datos” y “de control”.

Las instrucciones de acceso a memoria, como su nombre indica, son aquellas que realizan intercambios de datos con la memoria principal.

Las instrucciones de procesamiento de datos incluyen las operaciones aritméticas como sumar, restar, mover y comparar, y las operaciones lógicas como AND, OR y XOR.

Las instrucciones de control son aquellas que alteran el flujo normal del programa, modifican el valor del contador del programa.

## Segmentación

La segmentación permite que se procesen varias instrucciones al mismo tiempo en diferentes zonas o etapas del procesador. Las instrucciones son procesadas por todas las etapas y en cada una se procesa una parte de la instrucción.

En la imagen vemos cómo las instrucciones ocupan una única etapa y dejan el resto libres para otras instrucciones. Y también vemos cómo en cada ciclo avanzan por las diferentes zonas del procesador.

## Diseño

Aquí vemos el diseño final del procesador. Los diferentes bloques realizan tareas específicas, las líneas continuas (o negras) representan la ruta de datos, las líneas discontinuas (o azules) representan la ruta de control.

Las líneas verticales (o rojas) son los registros de control, estos son los elementos más importantes de la segmentación. En cada ciclo se actualizan sus valores, así se consigue que el hardware intermedio no tenga que permanecer ocioso y pueda procesar la siguiente instrucción.

Este procesador se ha segmentado en 5 etapas:

* 1. **Búsqueda de instrucción (IF):** Se carga la instrucción de memoria y se calcula la dirección de memoria de la siguiente.
  2. **Decodificación de la instrucción (ID):** Se preparan las señales de control y se cargan los operandos.
  3. **Ejecución (EX):** Se realiza la operación aritmético-lógica y se calcúla la dirección de salto si fuese necesario.
  4. **Memoria (MEM):** Se accede a la memoria de datos si fuese necesario.
  5. **Escritura (WB):** Se almacenan los resultados de la instrucción en el banco de registros.

# FALLOS

## Fallos (A)

Los fallos se pueden clasificar en fallos permanentes y fallos transitorios, para este proyecto nos centramos en el estos últimos.

## Fallos (B)

A su vez, dentro de esta categoría nos centramos en los fallos “Single-EventUpset”. Estos fallos afectan a los biestables invirtiendo su valor, pudiendo provocar un error en el sistema.

## Tolerancia

El método más común de tolerancia aplicado frente a estos fallos es la redundancia modular, que es el que se ha utilizado aquí.

Consiste en replicar un número impar de veces y conectar sus salidas a un nuevo elemento conocido como “Votador de mayoría” o simplemente “votador”.

El resultado de aplicar la redundancia modular triple a un “módulo A” cualquiera, produce el resultado que vemos.

## Votador

El votador puede implementarse de diferentes maneras, pero siempre debe satisfacer la tabla de verdad que vemos a la izquierda. A la derecha vemos un ejemplo de implementación, este es el votador que se ha utilizado en el proyecto.

## Votador(Ejemplo)

Para comprobar que el votador devuelve el resultado esperado veamos el siguiente ejemplo. El módulo conectado a la entrada C del votador ha sufrido un fallo, y por ello no coincide con los otros módulos. Como se puede observar el fallo se tolera, y la salida Z es equivalente a la mayoría de sus entradas (A y B).

# PRUEBAS

## Ejecución 1

El programa realiza una multiplicación mediante sumas acumulativas en un bucle.

Primero se accede a memoria, se lee un operando de la operación y se inicializan las demás variables. A continuación se ejecuta el bucle. Y para finalizar se almacena el resultado en memoria.

Vemos que el procesador ejecuta correctamente el programa

## Ejecución 2

Se vuelve a ejecutar el mismo programa en el procesador sin tolerancia. Esta vez se insertan tres fallos.

* 1. El primer fallo se genera en el bus de lectura de la memoria. Como consecuencia se carga un 4 en vez de un 5 en el registro R2.
  2. El segundo fallo se inserta en las señales de control de la etapa de memoria, esto provoca que se escriba en memoria un dato.
  3. El tercer fallo se genera en el bus del operando B de la etapa de ejecución. Esto provoca que la comparación dé un resultado positivo y se termine la ejecución del bucle antes de tiempo.

El resultado final es un programa que ha obtenido un resultado erróneo y además ha modificado datos en memoria que puede estar utilizando otro programa.

## Ejecución 3

Se ha ejecutado una última vez el programa sobre el procesador tolerante a fallos y se insertan los mismos fallos. Estos fallos se introducen en uno de los registros triplicados que almacenan la misma información.

Como podemos ver, el programa se ha ejecutado de forma correcta y sin modificar otros elementos en memoria.

## Recursos

La implementación del procesador utiliza un total de 1.952 registros, de los cuales 420 se utilizan par a los registros de control de las diferentes etapas. Mientras, la el procesador tolerante a fallos utiliza el triple para los registros de control (1260), el total asciende a 2.798.

La lógica combinacional del procesador se reparte en 1.015 LUTs, pero al insertar los votadores se utilizan 1.464 LUTs. La frecuencia de trabajo se reduce de 155 MHz a 146MHz debido a la inserción de la lógica de tolerancia.

## Conclusiones

Se ha implementado un microprocesador segmentado compatible con instrucciones ARM, y se le ha aplicado tolerancia a fallos.

Con las pruebas se ha demostrado que el microprocesador funciona correctamente y que la tolerancia le da un nivel más de fiabilidad. ¿Preguntas?