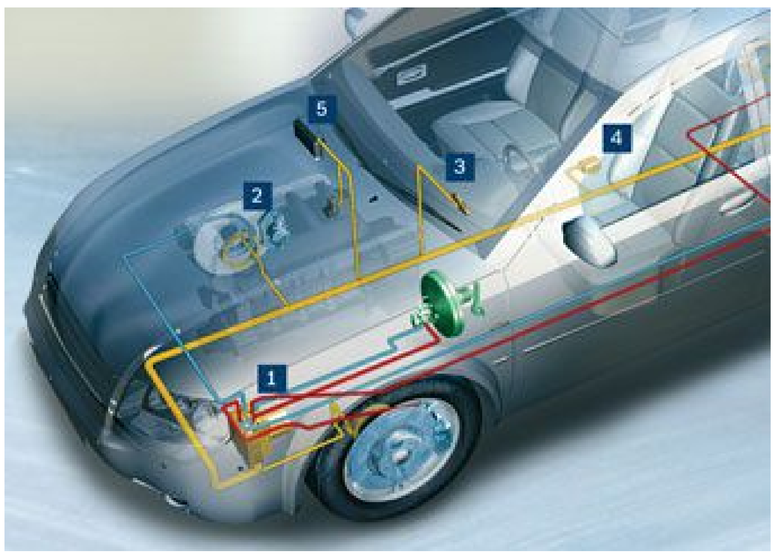
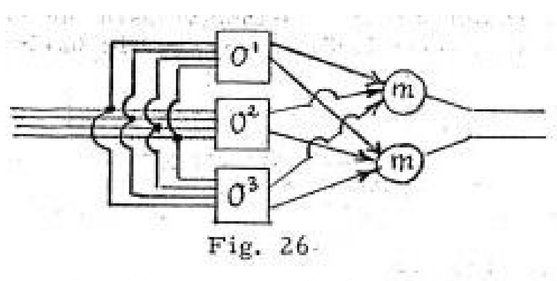
Tolerancia a fallas de sistemas embebidos en FPGA

Julio Pérez (julio@fing.edu.uy)
Dir. Académico: R. Canetti – Dir. Tesis: M. Sonza



El problema: las fallas y sus consecuencias

Sistemas de diferente tipo (un automóvil, un lavarropas, un programa para gestionar el funcionamiento de un banco) están expuestos a **fallas** (por errores de diseño, por roturas, etc.) que provocan efectos no deseados o impiden que el sistema cumpla correctamente con su función. En algunos sistemas esto no importa demasiado, en otros puede **costar mucho dinero** (el ejemplo del banco) o **poner en peligro la vida o la salud** de seres humanos (el automóvil, un marcapasos).



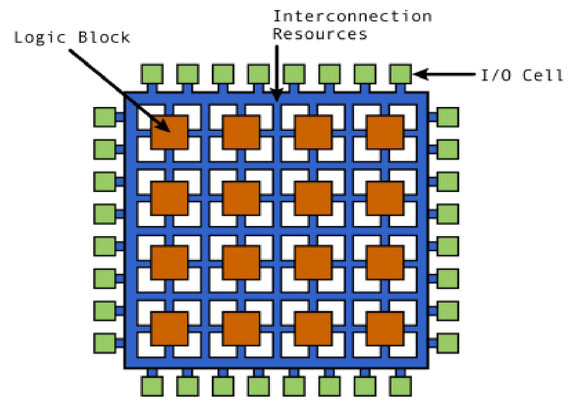
Von Neumann 1956

Los medios para atacarlo

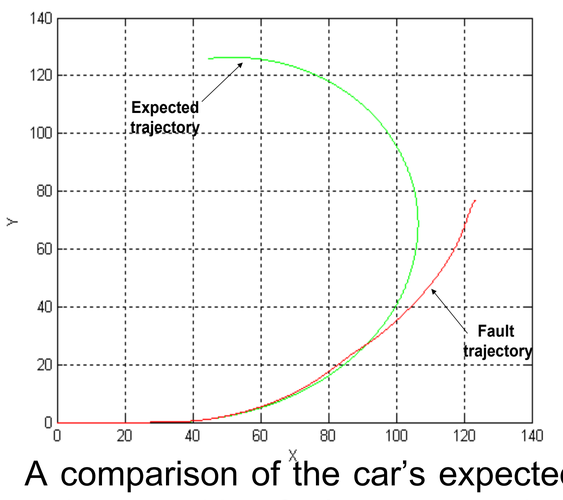
Existen numerosas técnicas para convivir con este problema: a) hacer las cosas de forma que haya la menor cantidad posible de fallas (**prevención**), b) detectar cuando ocurre una y tomar medidas para corregirla, c) que el sistema de un resultado correcto aunque haya alguna falla (**tolerancia**). En general estas técnicas implican algún tipo de **redundancia** (hacer las cuentas dos o tres veces y comparar, hacer la prueba del nueve, llevar una rueda auxiliar) que tienen algún **costo** asociado en **velocidad** de operación (si repito las cuentas hago menos cuentas por minuto), en **tamaño** del equipo que tengo que comprar (la rueda auxiliar) o en **energía** para que funcione el sistema (de nuevo la rueda).

En esta tesis

Se trabaja con técnicas para tolerancia a fallas para **determinado tipo de sistemas electrónicos** (sistemas embebidos en chips de lógica programable) y atacando a **un tipo de fallas en particular** (single event upsets o **SEUs**). Ese tipo de fallas son intermitentes y cambian el valor de datos almacenados en memoria. Son muy comunes en equipos que están expuestos a **radiación**, como un satélite o partes de una central nuclear, y pueden aparecer incluso en condiciones normales a medida que se reducen las dimensiones internas de los circuitos integrados.



¿Cómo saber si es suficiente?



A comparison of the car's expected trajectory with a faulty one provoked by one soft error in the memory elements of the Brake node

Ejemplo *no electrónico*: el *corrector* en un diario debe leer los textos que escriben otros (acá hay redundancia de recursos) y detectar faltas de ortografía. Es fácil saber cuántas faltas se detectan en este proceso, pero ¿cómo saber si el corrector hace bien su trabajo? Una forma es darle a corregir un texto al que agregamos faltas de ortografía (**inyección de fallas**) y ver cuántas de esas fallas inyectadas encuentra.

Durante la maestría (2003-2005) trabajé con técnicas de inyección de fallas en la red que comunica las unidades de **control electrónico de un automóvil**. Se comprobó que incluso con protocolos de red diseñados para ser robustos (bus CAN) una falla en uno de los nodos puede bloquear completamente la red con consecuencias graves para el vehículo.

UNIVERSIDAD DE LA REPUBLICA
FACULTAD DE INGENIERIA