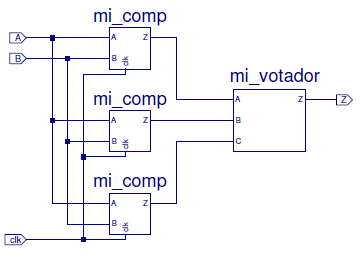
# TOLERANCIA A FALLOS TRANSITORIOS

Intro: nosotros solamente nos centamos en fallos tansitorios en este trabajo

## Como se consigue la tolerancia

Para evitar que los fallos transitorios se conviertan en errores y provoquen circunstancias imprevistas usaremos el método de “Triple modular Redundancy”.

El método aplicado consiste en triplicar el módulo, (o su funcionalidad modificando el circuito interno, pero manteniendo los resultados esperados), y utilizar un sistema votador para detectar, y si es posible enmascarar el fallo evitando que se produzca un error.



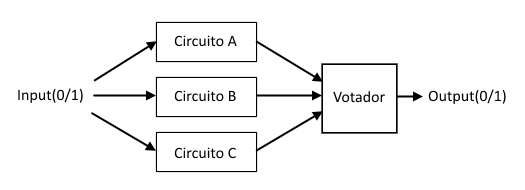


Figura 1. Sistema usando un Votador

El votador usado se conoce como “Majority Gate”. Como podemos ver en la “Figura 1”, recibe tres señales de tres módulos que aplican la misma función al mismo conjunto de valores. Las salidas de estos módulos deberían ser idénticas, salvo que se produzca un fallo en alguna puerta o biestable de los módulos. Este votador facilita la detección y la corrección de estos fallos del sistema.

## Votadores

Un votador es un sistema que recibe N entradas y tiene una salida que coincide con la mayoría de entradas.

Los votadores no están libres de fallos, ya que estos mismos pueden sufrir la radiación y provocarlos. Estos votadores están diseñados para intentar evitar un número limitado de fallos en los módulos repetidos dado por la función donde M es el número de fallos tolerados y N es el número de módulos repetidos, N debe ser un número impar.

La "Tabla de Verdad 1" muestra los valores de salida dependiendo de las entradas del Votador.

El "Mapa de Karnaught 1" es el correspondiente a la "Tabla de Verdad 1" y podemos obtener la función F(ABC) = AB + BC + AC para los valores de salida del votador.

La es el esquema del votador obtenido como resultado del diseño anterior.

//Dibujo de vtador 3 entradas de 1 bit

//Tala de verdad del votador, diagrama de karnaught e implementación final con puertas lógicas

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | F(ABC) |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Tabla de Verdad

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | AB |  |  |  |  |
| C |  | 00 | 01 | 11 | 10 |
|  | 0 | 0 | 0 | 1 | 0 |
|  | 1 | 0 | 1 | 1 | 1 |

Mapa de Karnaught

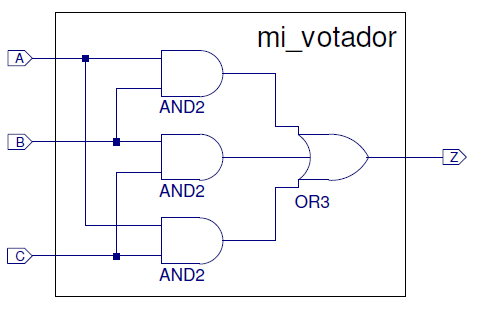
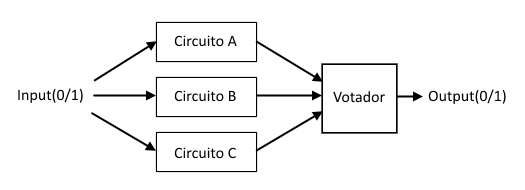
****

Figura 2. Esquema de votador

## Configuración del entorno Xilinx

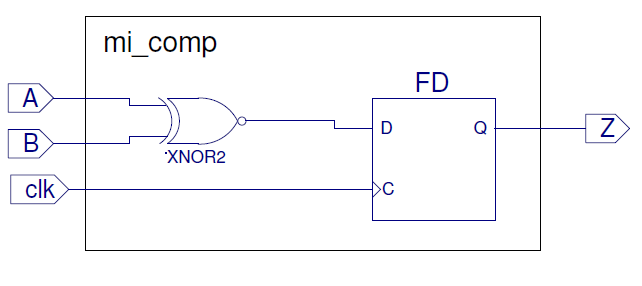
El software de Xilinx tiene por defecto una serie de opciones activadas para ayudar a que nuestro diseño ocupe la menor cantidad de silicio posible. Además de compactar y simplificar el diseño todo lo posible, también elimina partes innecesarias para el funcionamiento del mismo. Esto también incluye el hardware duplicado como en nuestro caso son los módulos triplicados.



Si dejamos que el Xilinx simplifique nuestra lógica obtendremos algo parecido a la

Nuestro objetivo es tener estos módulos varias veces y necesitamos que Xilinx no se tome la libertad de eliminar nuestro diseño. Para ello vamos a configurar el programa para que actúe como queremos.

TAREA



Implementas en vhdl un modulo sencillo con un bistble a la salida

modulo principal contiene 3 copias del modulo anterior conectados al votador

configuracion XILINX, busca como no eliminar elementos repetidos, poreque este diseño eliminaria los modulos repetidos y entonces tambien eliminaria elvotador

TAREA

## injeccion fallos

cada modulo sencillo tiene 3 lineas invertir salida, salida a 0, salida a 1

### MODO POST-INJECION

esto se hace mediante logica combinaional a la salida del biestable, para no introducir otro biestable que cambiaria la temporrizacion en etapas del procesador

### MODO PRE-INJECCION

en paralelo a la generacion de la entrada al biestable tambien funciona la lgica de injecciond efallo,

quizas el modo anterior introduce un retardo para las sieuigente etapa

este quizas iintroduce menos