Tipos de resets en una FPGA

Creador: David Rubio G.

Entrada: https://soceame.wordpress.com/2024/12/07/tipos-de-resets-en-una-fpga/

Blog: https://soceame.wordpress.com/

GitHub: https://github.com/DRubioG

Fecha última modificación: 23/02/2025

Esta entrada tiene la intención de mostrar los diferentes resets que puede y debe tener una FPGA tanto físicos como en firmware.

La FPGAs tienen como mínimo tres tipos de resets dependiendo de dónde vienen y quién lo provoca.

Reset de bitstream (Hard-reset)

Este reset es un tipo de reset específico de cada fabricante de FPGAs. Este reset tiene un pin específico de la FPGA, que es el reset más potente que tienen las FPGAs, porque es el reset que borra el bitstream de la lógica de la FPGA.

Este reset es un reset asíncrono, del que no existe una lógica que se puede ejecutar cuando se detecte, debido a que como borra el bitstream interno ninguna pieza del firmware puede hacer nada en ese instante, y al terminar se regraba el bitstream que había.
Salvo en las FPGAs de Microchip, que al ser de memoria Flash, el bitstream no se borra, por lo que este reset lo único que hace es volver a arrancar la lógica de la FPGA pero no el firmware, que se mantiene en el estado previo a la activación.

NOTA: *Estos resets hay que tener cuidado con ellos porque puede no haber garantías de que cuando se vuelva a cargar el bitstream en la FPGA el reset asíncrono, que es firmware, haya realizado su función de volver a un estado inicial.*

Este reset dependiendo del fabricante tiene varios nombres:

- *Xilinx*: **PROGRAM_B**
- *Intel (Altera)*: **nCONFIG**
- *Microchip (Microsemi)*: **DEVRST_N**
- *Lattice*: **PROGRAMN**

Todos estos resets, son resets activos a nivel bajo.

Reset asíncrono

Este es el reset más conocido, porque es el reset que permite volver al estado inicial, y es asíncrono porque **viene del mundo exterior**, del cuál se desconoce el sistema de reloj que tiene, suponiendo que lo tenga.

Este reset es el que se puede asignar a cualquier pin de la FPGA, porque no tiene un pin definido por el fabricante, si no que es el usuario el que lo tiene que configurar.

Este es un reset que se puede definir por firmware, por lo que realizará las tareas que especifique el firmware en este estado.

Este reset se recomienda que sea activo a nivel bajo *(¿Por qué? lo comento al final)*.

Reset síncrono

Este es un reset que se suele asociar con el reset asíncrono, estableciendo la relación de «*si no es un reset asíncrono, es un reset síncrono*«, y esto es un fallo muy común, porque esta señal está camuflada mediante un funcionamiento que es el mismo que el de una señal de reseteo, y además es la capacidad que tiene **el propio firmware** de volver a un estado inicial, sin la necesidad de un elemento externo que actúe mediante un reset.
De tal forma que puede haber un bloque FW que pueda resetear otro bloque FW de forma rápida y directa, y provocando este reset, el módulo vuelve al estado inicial.

NOTA: *todos los fabricantes recomiendan que exista un reset de este tipo. Aunque ciertamente aumente la complejidad del código, pero es fundamental tenerlo, además, se tiene que utilizar en las mismas condiciones del firmware en las que se utiliza el reset asíncrono.*

Este reset recibe diferentes nombres y tiene diferentes formas de activación:

- **enable** (en) [señal activa a nivel alto, reset activo a nivel bajo]
- **chip enable** (ce) [señal activa a nivel alto, reset activo a nivel bajo]
- **clear** (clr) [señal activa a nivel bajo, reset activo a nivel alto]
- **preset** (pst) [señal activa a nivel alto, reset activo a nivel bajo]

Entonces, todos los bloques firmware tienen que tener al menos dos formas de resets, aunque el reset síncrono nunca se utilice y se quede con un valor fijo *(esto ocurre cuando no hay un bloque superior que pueda provocar un reset de los diferentes bloques)*.

Además, esta señal tiene es una buena forma de facilitar la reusabilidad del firmware para diferentes proyectos.

NOTA: *yo para este reset recomiendo utilizar un reset activo a nivel, que sirva de contraparte al reset asíncrono.*

¿Por qué los reset externos tienen que ser activos a nivel bajo?

La respuesta es muy simple, porque la FPGA necesita una forma de saber que mundo exterior se ha *«ido a la porra»*.

Entonces, si el reset es *activo a nivel alto*, algo en el mundo exterior tiene que generar una señal a nivel alto, pero si la pista de la PCB(o cualquier otra cosa) que une la lógica externa con la FPGA se ha cortado, la FPGA nunca se va a poder resetear porque nadie se lo va a poder decir, por lo tanto se quedará viva de forma eterna, además, se pueden generar errores.
Sin embargo, si el reset es *activo a nivel bajo,* si se produce un fallo en la línea del reset provocado por cualquier cosa, la FPGA se enterará porque la línea se quedará sin alimentación, y entonces al caer a nivel bajo, el reset es automático y la FPGA se bloquea indicando que hay algo que no le gusta.

Por eso no se recomienda poner un pull-up en una línea de reset, porque si los dispositivos que atacan la línea de reset se caen, la FPGA nunca se enteraría porque vería siempre la línea a nivel

alto.
Solo se puede poner un pull-up en los pines de reset si no hay ningún dispositivo que ataque la línea o si se utiliza un reset manual con un botón.