Grandes mentiras de las FPGAs: los sistemas asíncronos

Creador: David Rubio G.

Entrada: https://soceame.wordpress.com/2024/12/17/grandes-mentiras-de-las-fpgas-los-sistemas-asincronos/

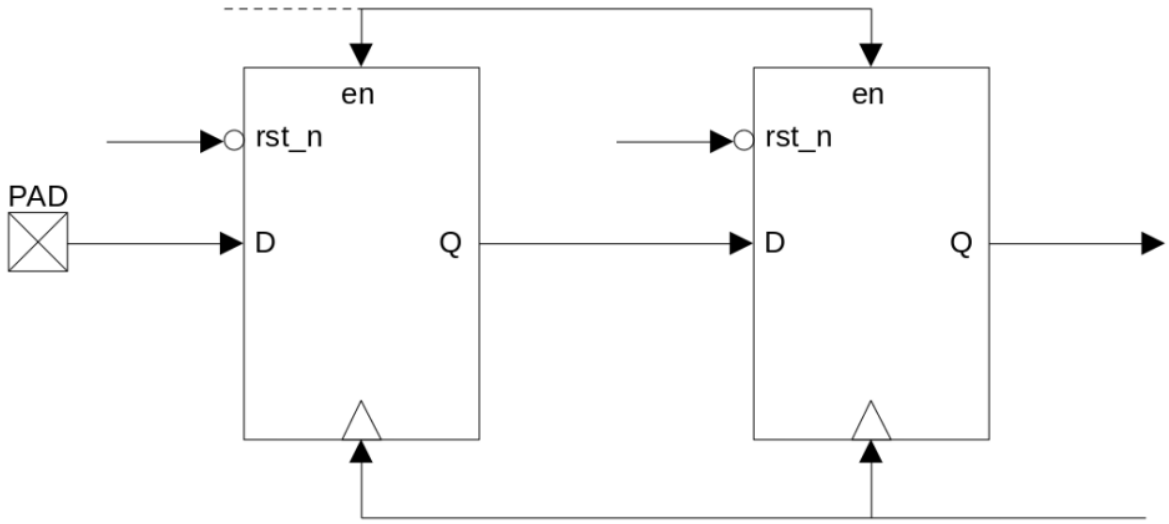
Blog: https://soceame.wordpress.com/

GitHub: https://github.com/DRubioG

Fecha última modificación: 23/02/2025

Una de las cosas que primero nos venden al empezar a programar FPGAs es que hay dos tipos de sistemas en una FPGA: los **sistemas síncronos**, que dependen de un reloj, y los **sistemas asíncronos**, que no dependen de un reloj.

Pero lo que no te cuentan es que los sistemas asíncronos pueden generar metaestabilidades dentro del sistema. Entonces, esos sistemas asíncronos se tienen que resincronizar con un reloj, por lo que dejan de ser asíncronos y pasan a ser síncronos.



Entonces podemos afirmar con bastante claridad que **los sistemas asíncronos en FPGAs no existen**, o mejor dicho, **no pueden existir en FPGAs**.

Entonces, ¿nos han estado engañando en las FPGAs?

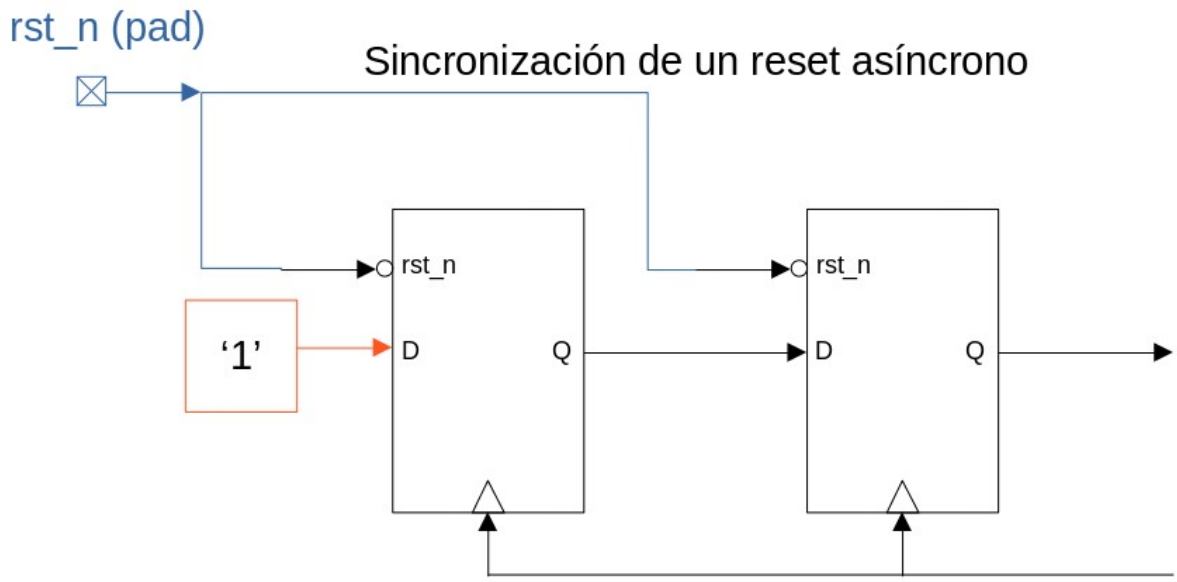
Mayoritariamente sí, pero no siempre.

Los sistemas asíncronos en FPGAs sí que pueden existir siempre que no haya un reloj de por medio, por lo que para sistemas **puramente combinacionales** sí que existen.
Pero para el resto de casos, que son la mayoría, en el momento que aparezca un reloj por algún sitio, todo el sistema tiene que ser síncrono, incluso puede haber un sistema con más de un sincronismo, porque puede tener más de un dominio de reloj interno.
Incluso si hay parte de un sistema combinacional, cuando aparece un reloj, el sistema combinacional tiene que cambiar a un sistema síncrono.

¿Y qué pasa con los resets?

El sistema asíncrono más básico que nos enseñan al empezar y que se mantiene así toda la vida es el sistema de reset asíncrono. Bien, pues al igual que todos los sistemas asíncronos se tiene que sincronizar antes de entrar.

El sincronismo del reset tiene una forma como la siguiente, donde queda fijado el valor del reset antes de empezar.



Y ahora viene la gran batalla con el reset, ¿cómo se tiene que utilizar el reset en todo el FW?

Todos sabemos que la condición del reset se coloca en un sitio u otro en función de si es asíncrono o no.

- **Modelo de reset asíncrono**

```
process(clk, rst_n, en)
begin
   if rst_n = '0' then
...
   elsif rising_edge(clk) then
      if en = '1' then
...
      elsif en = '0' then
...
      end if;
   en if;
end process;
```

- **Modelo de reset síncrono**

```
process(clk)
```

```
begin
   if rising_edge(clk) then
      if rst_n = '0' then
...
      elsif en = '1' then
...
      elsif en = '0' then
...
      end if;
   en if;
end process;
```

Estos son los dos modelos de estructurar el FW en función del tipo de reset. Entonces, como el reset es síncrono se tiene que utilizar el modelo de reset síncrono siempre…

(Aquí viene mi opinión personal, cualquiera puede opinar lo contrario que yo)

Yo aquí discrepo, debido a que si se utiliza el modelo de reset síncrono con el reset, tienes que esperar desde que salta un reset al menos 3 ciclos de reloj. Mientras que si se utiliza el modelo asíncrono el reset solo tendría que esperar dos ciclos de reloj, que son solo los que se necesita para sincronizarse. Además con el modelo síncrono, estarías resincronizando el reset cada vez que lo utilices.

Otra cosa que no estamos teniendo en cuenta es que la línea de reset pasa por lógica que es susceptible de modificar sus retardos por la temperatura de la FPGA, mientras que el reloj va por unas líneas que tienden a autocompensar los retardos, haciéndolos más estables. Entonces, el reset , a mi modo de ver, tiene que comenzar su operativa a la mayor brevedad y más si puede venir del mundo exterior.

Además, tengo que mantener el criterio con el que se diseño el reset asíncrono, porque el reset asíncrono ha sido creado así por alguna razón, si no hubiese sido creado síncrono.
Entonces, hay que distinguir el *reset síncrono*, creado así por una razón, y el *reset resincronizado*, que es un reset que ha sido creado asíncrono o en otro dominio de tiempo, y que por motivos de seguridad se ha vuelto a sincronizar en un nuevo dominio de tiempo, pero sigue manteniendo su funcionalidad inicial.

Y para terminar, muchos libros y ejemplos avanzados sobre desarrollos de FPGAs siguen utilizando el modelo de reset asíncrono, aunque resincronizan el reset a la entrada. Es verdad que algunos fabricantes como Xilinx recomiendan el uso del reset síncrono, porque reduce la lógica y permite generar zonas con reset más cortas. Pero tampoco comentan que cada bloque FW requeriría de al menos dos reset juntos pegados en el código, el reset resincronizado general y el reset síncrono específico del propio del bloque FW, que se podrían llegar a confundir porque su comportamiento es exactamente el mismo. Entonces, para evitar errores y porque la gente más entendida en esto lo utilizan, el modelo asíncrono es una opción.

Entonces, a mi parecer se puede seguir manteniendo el modelo de reset asíncrono para cualquier operación con el reset (que ha sido resincronizado), aunque se resincronice a la entrada de la FPGA.