Informe de Prestaciones

FIFO

Javier Presmanes Cardama

Samuel García Such

# Timing

En primer lugar, tras haber configurado el TimeQuest timing analyzer para funcionar a una frecuencia de 100MHz. Los resultados obtenidos son, para el peor de los casos:

Graphical user interface, application, Word

Description automatically generated

Siendo la frecuencia más baja de operación para nuestra FIFO de unos 180MHz estables.

# Flow Summary

Por otro lado, para saber cuánto ocuparía la FIFO en una FPGA, se ha compilado el proyecto obteniendo los siguientes datos :

Graphical user interface, application

Description automatically generated

De esta manera podemos asegurar que el número total de registros utilizados en el diseño es de 26, utilizando un total de 23 ALMs (adaptative logic modules), siendo este <1% de la capacidad que tiene la FPGA

# Testing

Para la verificación del dispositivo se han generado los siguientes casos de prueba:

Text

Description automatically generated

## test\_reset

Comprueba que el sistema se reinicia asíncronamente y que el estado pasa a EMPTY

## test\_clear

Comprueba que el Sistema se reinicia síncronamente y que el estado pasa a EMPTY

## test\_force\_only\_write

Comprueba que al escribir sin leer se pasa de estado EMPTY a OTHER y que el registro del contador de escritura crece

## test\_write\_and\_read

Comprueba que, al escribir y leer a la vez, la salida es igual a la entrada y que ningún registro de lectura o escritura es implementado, asimismo, comprueba que el estado es EMPTY

## test\_use\_dw

Comprueba que al escribir y leer varios elementos el valor use\_dw va cambiando

## test\_overflow

Comprueba que cuando intentamos escribir más elementos de los que permite la FIFO, esta deja de escribir elementos nuevos porque ya se encuentra llena.

## test\_read\_full\_fifo

Comprueba que cuando la FIFO está llena y la vacías, se vacía realmente, comprobando para ellos los valores del USE\_DW y el estado actual

## test\_change\_state\_empty\_other

Comprueba que el estado actual de la FIFO es EMPTY, luego escribe un elemento y comprueba que es OTHER

## test\_change\_state\_other\_empty

Comprueba que el estado actual de la FIFO es OTHER, lee un elemento y comprueba que es EMPTY (teniendo en cuenta que solo hay un elemento en la FIFO)

## test\_change\_state\_other\_full

Escribe 31 elementos en la FIFO, comprueba que esta en estado OTHER, vuelve a escribir otro elemento y comprueba que está en estado FULL.

## test\_change\_state\_full\_other

Escribe 32 elementos en la FIFO, comprueba que esta en estado FULL, lee un elemento y comprueba que está en estado OTHER.

# Conclusión

Tras la realización de todos los test con aserciones he obtenido el siguiente resultado con QuestaSim :

A picture containing text

Description automatically generatedDonde se puede observar el tiempo en el que se ha ejecutado la prueba, que aserciones se han comprobado y si ha habido algún error.

Estos datos pueden corroborarse con la información sobre los asserts que te proporciona QuestaSim :

Table

Description automatically generated

Donde se ve claramente el porcentaje de prueba que han sido pasados por el banco de pruebas.

Basándonos en los resultados obtenidos, creemos firmemente que la FIFO cumple con su cometido de manera correcta, aunque para una mayor precisión deberíamos de haber utilizado un sistema de asserts con properties de manera que habríamos podido validar también la parte del diseño y nos solo de la verificación.