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:

	Fmax	Restricted Fmax	Clock Name	Note
1	250.19 MHz	180.02 MHz	fifo_iff.CLK	limit due to minimum period restriction (tmin)

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 :

Flow Status	Successful - Sun Oct 10 13:35:28 2021
Quartus Prime Version	17.1.0 Build 590 10/25/2017 SJ Lite Edition
Revision Name	FIFO_32_8
Top-level Entity Name	FIFO_32_8
Family	Cyclone V
Device	5CGXFC7C7F23C8
Timing Models	Final
Logic utilization (in ALMs)	23 / 56,480 (< 1 %)
Total registers	26
Total pins	28 / 268 (10 %)
Total virtual pins	0
Total block memory bits	256 / 7,024,640 (< 1 %)
Total DSP Blocks	0 / 156 (0 %)
Total HSSI RX PCSs	0 / 6 (0 %)
Total HSSI PMA RX Deserializers	0 / 6 (0 %)
Total HSSI TX PCSs	0 / 6 (0 %)
Total HSSI PMA TX Serializers	0 / 6 (0 %)
Total PLLs	0 / 13 (0 %)
Total DLLs	0 / 4 (0 %)

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:

```
task run_battery_test();
    battery_test_display("run_tests_change_state");

    // Bsaic tests
    test_reset();
    test_clear();
    test_force_only_write();
    test_write_and_read();
    test_use_dw();
    test_overflow();
    test_read_full_fifo();

    // Change of states
    test_change_state_empty_other();
    test_change_state_other_empty();
    test_change_state_other_full();
    test_change_state_full_other();
endtask
```

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.

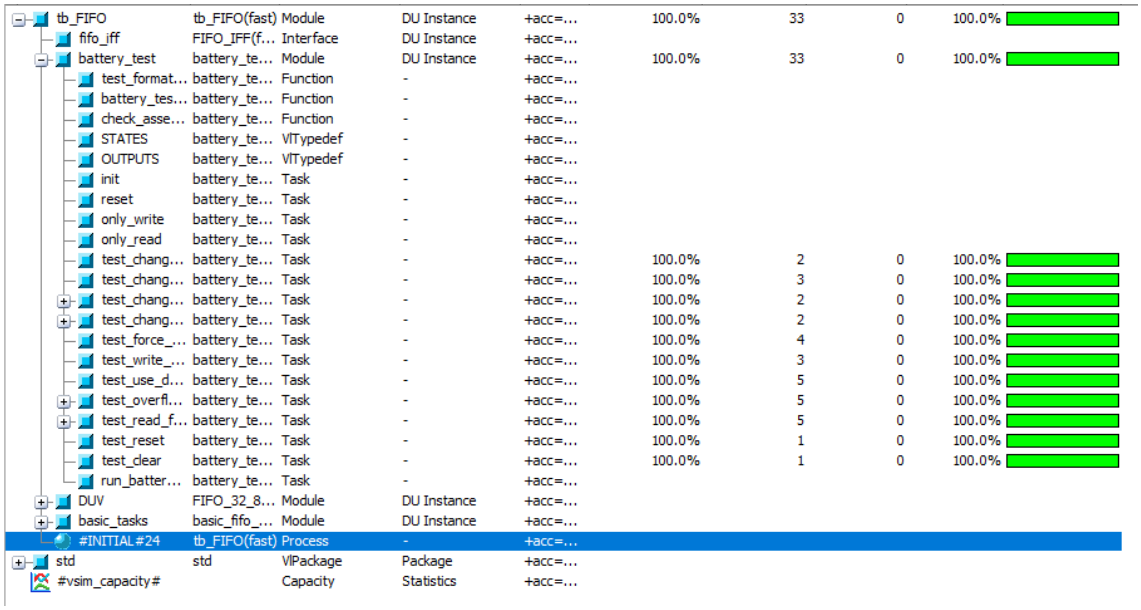
Conclusión

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

```
# ========== [ run_tests_change_state ] ==========
# 0.020ns || > [TEST] test_reset
# 0.032ns || > > [CHECK ASSERTION] NEW STATE == EMPTY
# 0.032ns || > [TEST] test_clear
# 0.071ns || > > [CHECK ASSERTION] NEW STATE == EMPTY
# 0.071ns || > [TEST] test_force_only_write
# 0.140ns || > > [CHECK ASSERTION] DATA_OUT == DATA_IN
# 0.140ns || > > [CHECK ASSERTION] COUNTER_W INCREMENTED
# 0.140ns || > > [CHECK ASSERTION] COUNTER_W INCREMENTED
# 0.140ns || > > [CHECK ASSERTION] NEW STATE == OTHER
# 0.140ns || > [TEST] test_write_and_read
# 0.180ns || > > [CHECK ASSERTION] DATA_OUT == DATA_IN
# 0.180ns || > > [CHECK ASSERTION] NEX STATE == EMPTY
# 0.180ns || > > [CHECK ASSERTION] OUTPUT SELECTOR == OUT_IS_IN
# 0.180ns || > [TEST] test_use_dw
# 0.220ns || > > [CHECK ASSERTION] USE_DW == 1
# 0.240ns || > > [CHECK ASSERTION] USE_DW == 2
# 0.260ns || > > [CHECK ASSERTION] USE_DW == 3
# 0.280ns || > > [CHECK ASSERTION] USE_DW == 2
# 0.300ns || > > [CHECK ASSERTION] USE_DW == 1
# 0.300ns || > [TEST] test_overflow
# 0.312ns || > > [CHECK ASSERTION] CURRENT_STATE == EMPTY
# Written 40 elements on FIFO
# 1.140ns || > > [CHECK ASSERTION] CURRENT_STATE == FULL
# 1.140ns || > > [CHECK ASSERTION] USE_DW == 0
# 1.160ns || > > [CHECK ASSERTION] CURRENT_STATE == OTHER
# 1.160ns || > > [CHECK ASSERTION] USE_DW == 31
# 1.160ns || > [TEST] test_read_full_fifo
# 1.172ns || > > [CHECK ASSERTION] CURRENT_STATE == EMPTY
# Written 40 elements on FIFO
# 2.000ns || > > [CHECK ASSERTION] CURRENT_STATE == FULL
# 2.000ns || > > [CHECK ASSERTION] USE_DW == 0
# 2.820ns || > > [CHECK ASSERTION] CURRENT_STATE == EMPTY
# 2.820ns || > > [CHECK ASSERTION] USE_DW == 0
# 2.820ns || > [TEST] test_change_state_empty_other
# 2.852ns || > > [CHECK ASSERTION] CURRENT STATE == EMPTY
# 2.881ns || > > [CHECK ASSERTION] CURRENT STATE == OTHER
# 2.881ns || > [TEST] test_change_state_other_empty
# 2.912ns || > > [CHECK ASSERTION] CURRENT STATE == EMPTY
# 2.940ns || > > [CHECK ASSERTION] CURRENT STATE == OTHER
# 2.960ns || > > [CHECK ASSERTION] CURRENT STATE == EMPTY
# 2.960ns || > [TEST] test_change_state_other_full
# 3.620ns || > > [CHECK ASSERTION] CURRENT STATE == OTHER
# 3.640ns || > > [CHECK ASSERTION] CURRENT STATE == FULL
# 3.640ns || > [TEST] test_change_state_full_other
# 4.340ns || > > [CHECK ASSERTION] CURRENT STATE == FULL
# 4.360ns || > > [CHECK ASSERTION] CURRENT STATE == OTHER
```

Donde 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 :



tb_FIFO	tb_FIFO(fast)	Module	DU Instance	+acc=...	100.0%	33	0	100.0%
fifo_iff	FIFO_IFF(f...	Interface	DU Instance	+acc=...				
battery_test	battery_te...	Module	DU Instance	+acc=...	100.0%	33	0	100.0%
test_format...	battery_te...	Function	-	+acc=...				
battery_tes...	battery_te...	Function	-	+acc=...				
check_asse...	battery_te...	Function	-	+acc=...				
STATES	battery_te...	VlTypedef	-	+acc=...				
OUTPUTS	battery_te...	VlTypedef	-	+acc=...				
init	battery_te...	Task	-	+acc=...				
reset	battery_te...	Task	-	+acc=...				
only_write	battery_te...	Task	-	+acc=...				
only_read	battery_te...	Task	-	+acc=...				
test_chang...	battery_te...	Task	-	+acc=...	100.0%	2	0	100.0%
test_chang...	battery_te...	Task	-	+acc=...	100.0%	3	0	100.0%
test_chang...	battery_te...	Task	-	+acc=...	100.0%	2	0	100.0%
test_chang...	battery_te...	Task	-	+acc=...	100.0%	2	0	100.0%
test_force_...	battery_te...	Task	-	+acc=...	100.0%	4	0	100.0%
test_write_...	battery_te...	Task	-	+acc=...	100.0%	3	0	100.0%
test_use_d...	battery_te...	Task	-	+acc=...	100.0%	5	0	100.0%
test_overfl...	battery_te...	Task	-	+acc=...	100.0%	5	0	100.0%
test_read_f...	battery_te...	Task	-	+acc=...	100.0%	5	0	100.0%
test_reset	battery_te...	Task	-	+acc=...	100.0%	1	0	100.0%
test_clear	battery_te...	Task	-	+acc=...	100.0%	1	0	100.0%
run_batter...	battery_te...	Task	-	+acc=...				
DUV	FIFO_32_8...	Module	DU Instance	+acc=...				
basic_tasks	basic_fifo_...	Module	DU Instance	+acc=...				
#INITIAL#24	tb_FIFO(fast)	Process	-	+acc=...				
std	std	VlPackage	Package	+acc=...				
#vsim_capacity#		Capacity	Statistics	+acc=...				

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.