Progetto Reti Logiche

Gabriele Giannotto

2022/2023

Contents

1 Introduzione **1**
1.1 Esempio con analisi 2

2 Architettura **4**
2.1 Interfaccia del Componente 4
2.2 Macchina a stati e datapath 5
2.2.1 Macchina a stati 5
2.2.2 Datapath 7
2.3 Memoria RAM 9

3 Risultati sperimentali **10**
3.1 Sintesi 10
3.2 Simulazioni 12

4 Conclusioni **14**

5 Appendice-SCHEMATIC **14**

1 Introduzione

Il progetto di quest'anno prevede di implementare un modulo Hardware(in VHDL) che si interfacci ad una memoria RAM sincrona e ne restituisca il risultato. La specifica di questo modulo è la seguente:

Il modulo deve fornire il dato letto da memoria in uscita attraverso uno dei 4 canali disponibili. L'indirizzo di memoria e il canale di uscita vengono ricevuti attraverso un ingresso seriale a 1 bit W letto sul fronte di salita del clock quando il segnale START è alto. Il modulo può comprendere quali siano i bit associati al canale e quali all'indirizzo di memoria. L'utente può leggere in parallelo tutti e 4 i canali solo per un ciclo di clock (questo ciclo viene identificato grazie al segnale DONE {vedi seguito}), quindi si presuppone che i valori che

dovranno essere presentati sui canali vadano salvati. Il modulo è regolato dal clock e da un segnale di reset, che riporta la macchina nello stato di partenza.

1.1 Esempio con analisi

Qui di seguito si può vedere un esempio di come il modulo debba funzionare.

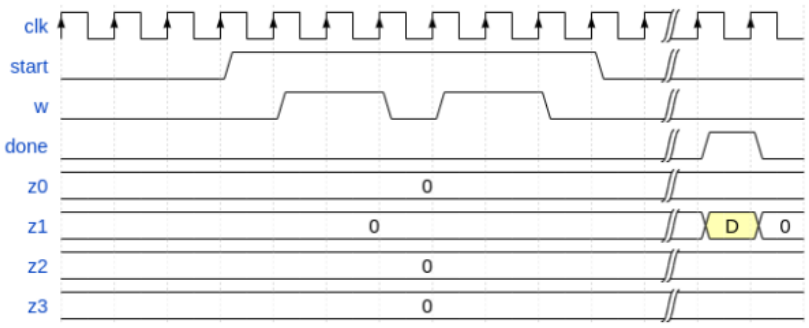

Figure 1: esempio di funzionamento 1: lettura del dato dall'indirizzo di memoria 0000000000010110 e scrittura sul canale Z1 (01).

Interfacce del modulo: Il modulo ha due ingressi primari da 1 bit (W e START) e 5 uscite primarie. Le uscite sono le seguenti: quattro da 8 bit (Z0, Z1, Z2, Z3) e una da 1 bit (DONE). Inoltre, il modulo ha un segnale di clock CLK, unico per tutto il sistema e un segnale di reset, RESET, anch'esso unico.

Acquisizione canale di uscita(2 bit)(AZIONE 1): quando il segnale START(i_start) diventa 1 si inizia a leggere il canale di uscita.
Z0 ->00
Z1 ->01
Z2 ->10
Z3 ->11

Acquisizione indirizzo di memoria(16 bit)(AZIONE 2): dopo aver letto il canale di uscita, se il segnale START è ancora alto, si inizia a leggere l'indirizzo di memoria. Nel caso in qui il numero di cicli di clock, in cui il segnale start è alto, fosse inferiore a 16, l'indirizzo viene esteso con degli 0 per raggiungere un indirizzo a 16 bit. Nel caso il segnale START, dopo aver letto il canale, non rimanesse alto, l'indirizzo di memoria sarebbe pari a 16 bit uguali a zero.

(N = 7) 1010111 –>0000000001010111
(N = 16) 1110000001010111 –>1110000001010111
(N = 0) 0000000000000000 –>0000000000000000

Validità input: La sequenza di ingresso è valida quando il segnale START è alto (1) e termina quando il segnale START è basso (0). Il segnale START rimane alto per almeno di 2 cicli di clock e al massimo per 18 cicli di clock (2 bit del canale e 16 bit per il massimo numero di bit per l'indirizzo).

Scrittura canale di uscita(AZIONE 3): Le uscite Z0(o_z0),..., Z3(o_z3) sono inizialmente 0. I valori sono visibili solo quando il valore di DONE(o_done) è 1. Quando il segnale DONE è 0 tutti i canali Z0, Z1, Z2 e Z3 devono essere posti a zero. Quando DONE=1 il canale letto in ingresso assumerà il valore letto da memoria attraverso l'indirizzo letto in ingresso e gli altri canali assumeranno l'ultimo valore che era stato scritto su di essi.

Dettagli sul segnale DONE: Quando lo specifico canale viene scritto, il segnale DONE passa da 0 passa a 1 e rimane attivo per un solo ciclo di clock.

Dettagli sul segnale START: Il segnale START è garantito rimanere a 0 fino a che il segnale DONE non è tornato a 0.

Dettagli sul segnale RESET: Prima del primo START=1 (e prima di richiedere il corretto funzionamento del modulo) verrà sempre dato il RESET(i_rst) (RESET=1). Una possibile successiva elaborazione con START=1 non dovrà attendere il reset del modulo. Ogni qual volta viene dato il segnale di RESET (RESET=1), il modulo viene re-inizializzato)(il segnale di reset è quindi asincrono).

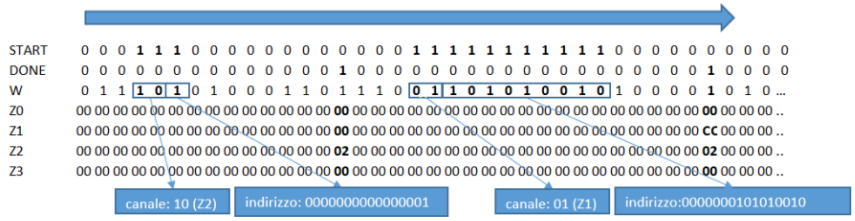

Figure 2: esempio di funzionamento 2

2 Architettura

Qui di seguito verrà analizzata l'architettura di questo progetto. La struttura del progetto si divide in una macchina a stati ed un datatpath.

2.1 Interfaccia del Componente

Il modulo con cui ci si può interfacciare ha la seguente interfaccia:

```
entity project_reti_logiche is
    port (
        i_clk   : in std_logic;
        i_rst   : in std_logic;
        i_start : in std_logic;
        i_w     : in std_logic;

        o_z0    : out std_logic_vector(7 downto 0);
        o_z1    : out std_logic_vector(7 downto 0);
        o_z2    : out std_logic_vector(7 downto 0);
        o_z3    : out std_logic_vector(7 downto 0);
        o_done  : out std_logic;

        o_mem_addr : out std_logic_vector(15 downto 0);
        i_mem_data : in std_logic_vector(7 downto 0);
        o_mem_we   : out std_logic;
        o_mem_en   : out std_logic
    );
end project_reti_logiche;
```

Figure 3: Interfaccia componente

-) i_clk è il segnale di CLOCK in ingresso generato dal Test Bench
-) i_rst è il segnale di RESET che inizializza la macchina pronta per ricevere il primo segnale di START 3
-) i_start è il segnale di START generato dal Test Bench
-) i_w è il segnale W precedentemente descritto e generato dal Test Bench
-) o_z0, o_z1, o_z2, o_z3 sono i quattro canali di uscita
-) o_done è il segnale di uscita che comunica la fine dell'elaborazione
-) o_mem_addr è il segnale (vettore) di uscita che manda l'indirizzo alla memoria
-) i_mem_data è il segnale (vettore) che arriva dalla memoria in seguito ad una richiesta di lettura
-) o_mem_en è il segnale di ENABLE da dover mandare alla memoria per poter comunicare (sia in lettura che in scrittura)

-) o_mem_we è il segnale di WRITE ENABLE da dover mandare alla memoria (=1) per poter scriverci. Per leggere da memoria esso deve essere 0

2.2 Macchina a stati e datapath

Qui di seguito verrà analizzata la macchina a stati e il datapath del modulo.

2.2.1 Macchina a stati

La macchina a stati, nello specifico di Moore, è composta da 8 stati(S1,S2,...,S8) e usa come ingressi per gli stati i_start e i_rst. In VHDL, per poter rappresentare questa macchina a stati, si è creato un modulo project_reti_logiche dove al suo interno vi sono 3 process(sono 3 per facilitare la visualizzazione del codice) e un component:
1) Il primo process, attivato dal fronte di salita del segnale clock oppure dal segnale alto del segnale reset, gestisce il cambio di stato.
2) Il secondo process, attivato dal cambio di stato oppure dal segnale start, gestisce la risoluzione del nuovo stato.
3) Il terzo process, attivato dal cambio di stato , gestisce il cambio delle variabili interne allo stato, ovvero i segnali che devono essere posti a 1/0 per far funzionare il modulo.
4) Il component si riferisce al modulo datapath che verrà analizzato di seguito.

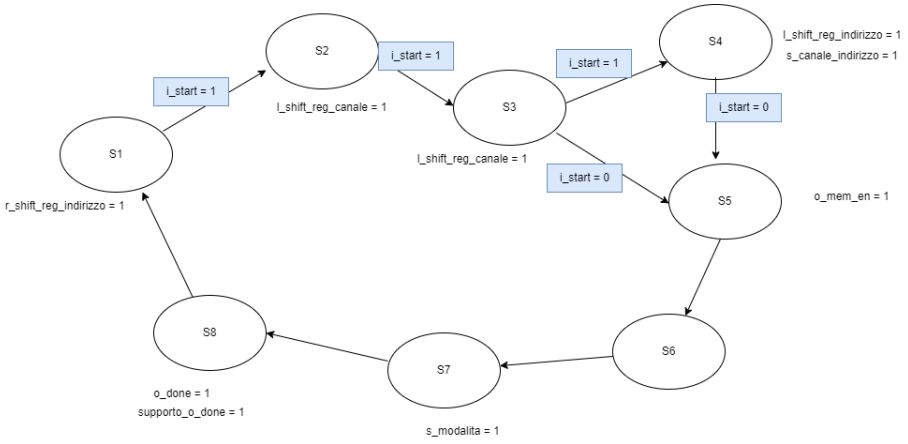

Figure 4: Macchina a stati. Nota 1: sul grafico non sono segnati i segnali clock e reset, guardare la descrizione qui sotto per analisi dettagliata. Nota2: Questa macchina sarebbe collegata al grafico del datapath, ma per favorire la comprensione si è deciso di analizzarli separatamente.

Stato S1: Questo è lo stato iniziale del modulo. Qui viene settato a 1 il segnale r_shift_reg_inditizzo che serve per resettare i valori all'interno del registro che contiene l'indirizzo di memoria. Questo segnale serve perché, nel caso venisse letto un secondo indirizzo, non vengano a crearsi problemi sui bit dato che l'indirizzo letto non è obbligatoriamente di 16 bit. Gli altri segnali vengono posti nella loro condizione di riposo.

Stato S2: Questo è lo stato in cui si legge il primo bit che indica il canale di uscita da scrivere. Per fare ciò viene posto a 1 il segnale l_shift_reg_canale. Gli altri segnali vengono posti nella loro condizione di riposo.

Stato S3: Questo è lo stato in cui si legge il secondo bit che indica il canale di uscita da scrivere. Per fare ciò viene posto a 1 il segnale l_shift_reg_canale. Gli altri segnali vengono posti nella loro condizione di riposo.

Stato S4: Questo è lo stato in cui si legge l'indirizzo di memoria. Per fare ciò viene posto a 1 il segnale l_shift_reg_indirizzo e il segnale s_canale_indirizzo. Gli altri segnali vengono posti nella loro condizione di riposo.

Stato S5: Questo è lo stato in cui si abilita la lettura in memoria. Per fare ciò viene posto a 1 il segnale o_mem_en .Gli altri segnali vengono posti nella loro condizione di riposo.

Stato S6: Questo è lo stato in cui si legge da memoria.Questo stato è necessario dato che la RAM utilizzata è sincrona e quindi vi è la necessità di uno stato di appoggio.Gli altri segnali vengono posti nella loro condizione di riposo.

Stato S7: Questo è lo stato in cui viene scritto il canale con il valore letto da memoria.Per fare ciò viene posto a 1 il segnale s_modalita .Gli altri segnali vengono posti nella loro condizione di riposo.

Stato S8: Questo è lo stato in cui vengono fatti visualizzare i valori dei canali per l'utente esterno. Per fare ciò viene posto a 1 il segnale o_done e supporto_o_done. Gli altri segnali vengono posti nella loro condizione di riposo.

Informazioni cambi di stato: Il cambio di stato può avvenire ad ogni ciclo di clock(un cambio per ciclo). La transizione avviene in automatico se essa non ha delle condizioni su segnali di ingresso, altrimenti devono essere verificate le suddette condizioni.

Informazioni su reset: Nel caso il segnale di reset venga posto ad 1 lo stato diventa S1 e ,quindi, tutti i segnali vengono inizializzati alla condizione di partenza del modulo ed i registri vengono azzerati.

Informazioni su condizione di riposo dei segnali di controllo: queste sono le condizioni di riposo dei segnali:
l_shift_reg_indirizzo –>0
s_canale_indirizzo –>0
r_shift_reg_indirizzo –>0
l_shift_reg_canale –>0
o_mem_we –>0
o_mem_en –>0
o_done –>0
supporto_o_done –>0
s_modalita –>0

2.2.2 Datapath

Questo Datapath contiene le componentistiche atte a soddisfare l'obbiettivo del processo.In VHDL, per poter rappresentare questo datapath, si è creato un modulo datapath dove al suo interno vi sono i process per gestire i registri(uno per registro, per leggibilità del codice) e dei with select per gestire i MUX e DEMUX del modulo.
Qui di seguito verranno analizzati i segali di ingresso al datapath, i registri e i MUX/DEMUX:

l_shift_reg_indirizzo: segnale utilizzato per abilitare la scrittura del registro shift_registro_indirizzo, registro per la scrittura dell'indirizzo di memoria.

s_canale_indirizzo: segnale utilizzato per pilotare il DEMUX canale_indirizzo, utilizzato per inviare il bit di informazione al registro corretto.

r_shift_reg_indirizzo: segnale utilizzato per azzerare il valore del registro shift_registro_indirizzo.

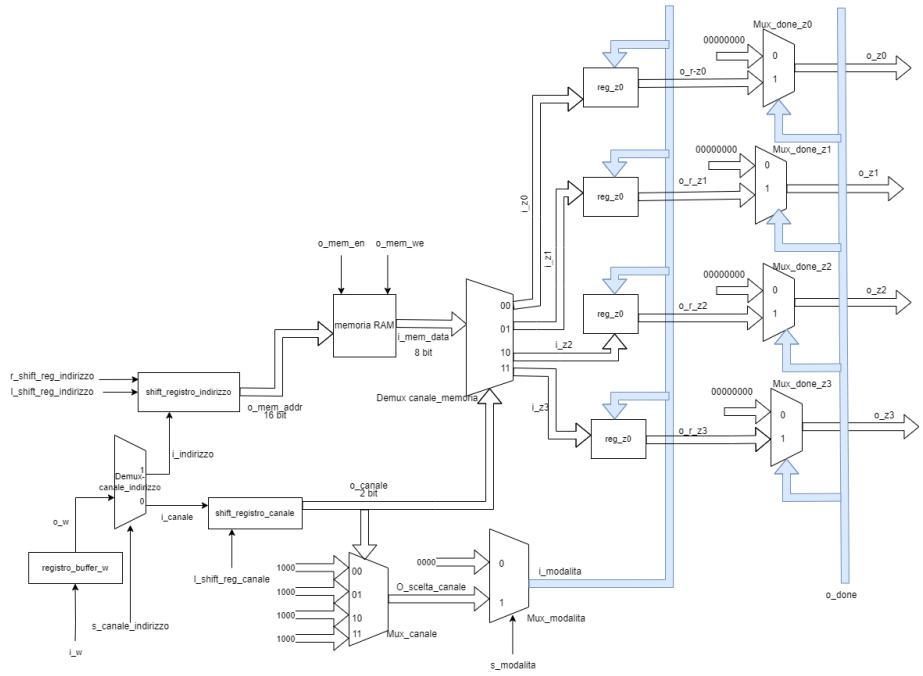

Figure 5: Datapath.Nota 1: sul grafico non sono segnati i segnali clock e reset, guardare la descrizione qui sotto per analisi dettagliata.

l_shift_reg_canale: segnale utilizzato per abilitare la scrittura del registro shift_registro_canale, registro per la scrittura del canale di uscita.

o_mem_we: segnale per scegliere lettura/scrittura in memoria(scrittura −>1)

o_mem_en: segnale per abilitare lettura/scrittura in memoria(abilitare −>1)

o_done: segnale che pilota i MUX Mux_done_z0,...,Mux_done_z3, mux che servono per mostrare in uscita i valori associati ai canali oppure 8 bit pari a zero. Per utilizzare questo segnale si è dovuto aggiungere supporto_o_done dato che o_done è un segnale di output.

s_modalita: segnale che pilota il MUX Mux_modalita, mux che serve per scegliere tra la scrittura di uno specifico canale e la trasparenza dei canali.

Mux_canale: mux che serve per settare il segnale o_scelta_canale, segnale che servirà per la possibile scrittura sul canale.

Registri shift: i registri shift_registro_indirizzo e shift_registro_canale sono dei registri che shiftano a sinistra con partenza a destra.

Demux_canale_memoria: mux che serve per indirizzare il valore letto da memoria al registro corretto.

registro_buffer_w: registro di utilità per poter gestire lettura sul fronte di salita del segnale i_w .

2.3 Memoria RAM

La memoria e il suo protocollo può essere estratto dalla seguente descrizione VHDL che fa parte del test bench e che è derivata dalla User guide di VIVADO disponibile al seguente link: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_3/ug901-vivado-synthesis.pd

```
-- Single-Port Block RAM Write-First Mode (recommended template)
--
-- File: rams_02.vhd
--
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity rams_sp_wf is
port(
  clk  : in  std_logic;
  we   : in  std_logic;
  en   : in  std_logic;
  addr : in  std_logic_vector(15 downto 0);
  di   : in  std_logic_vector(7 downto 0);
  do   : out std_logic_vector(7 downto 0)
);
end rams_sp_wf;

architecture syn of rams_sp_wf is
type ram_type is array (65535 downto 0) of std_logic_vector(7 downto 0);
signal RAM : ram_type;
begin
  process(clk)
    begin
    if clk'event and clk = '1' then
      if en = '1' then
        if we = '1' then
          RAM(conv_integer(addr)) <= di;
          do                      <= di after 2 ns;
        else
          do <= RAM(conv_integer(addr)) after 2 ns;
        end if;
      end if;
    end if;
  end process;
end syn;
```

Figure 6: RAM

3 Risultati sperimentali

Ora verrà analizzato il testing del modulo.

3.1 Sintesi

Qui di seguito possiamo notare i report di sintesi.

report_timing: possiamo notare, dato un clock di 100ns, che lo slack è circa 97 sintomo che il modulo riesce a svolgere le operazione nel ciclo di clock da

100ns e che potrebbe funzionare con un clock minore.

report_utilization: possiamo notare che non vi sono Latch in questo progetto e quindi i cambi che avvengono nel modulo sono ben temporizzati.

Location	Delay type	Incr(ns)	Path(ns)		Netlist Resource(s)
	(clock clock rise edge)	0.000	0.000	r	
		0.000	0.000	r	i_clk (IN)
	net (fo=0)	0.000	0.000		i_clk
	IBUF (Prop_ibuf_I_O)	0.944	0.944	r	i_clk_IBUF_inst/O
	net (fo=1, unplaced)	0.800	1.744		i_clk_IBUF
	BUFG (Prop_bufg_I_O)	0.096	1.840	r	i_clk_IBUF_BUFG_inst/O
	net (fo=54, unplaced)	0.584	2.424		i_clk_IBUF_BUFG
	FDCE			r	FSM_sequential_cur_state_reg[1]/C
	FDCE (Prop_fdce_C_Q)	0.456	2.880	r	FSM_sequential_cur_state_reg[1]/Q
	net (fo=60, unplaced)	0.893	3.773		DATAPATH0/cur_state[1]
	LUT5 (Prop_lut5_I0_O)	0.295	4.068	r	DATAPATH0/eqOp/O
	net (fo=8, unplaced)	0.511	4.579		DATAPATH0/eqOp__0
	FDCE			r	DATAPATH0/o_r_z0_reg[0]/CE
	(clock clock rise edge)	100.000	100.000	r	
		0.000	100.000	r	i_clk (IN)
	net (fo=0)	0.000	100.000		i_clk
	IBUF (Prop_ibuf_I_O)	0.811	100.811	r	i_clk_IBUF_inst/O
	net (fo=1, unplaced)	0.760	101.570		i_clk_IBUF
	BUFG (Prop_bufg_I_O)	0.091	101.661	r	i_clk_IBUF_BUFG_inst/O
	net (fo=54, unplaced)	0.439	102.100		DATAPATH0/CLK
	FDCE			r	DATAPATH0/o_r_z0_reg[0]/C
	clock pessimism	0.178	102.279		
	clock uncertainty	-0.035	102.243		
	FDCE (Setup_fdce_C_CE)	-0.202	102.041		DATAPATH0/o_r_z0_reg[0]
	required time		102.041		
	arrival time		-4.579		
	slack		97.463		

Figure 7: report_timing

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	34	0	0	134600	0.03
LUT as Logic	34	0	0	134600	0.03
LUT as Memory	0	0	0	46200	0.00
Slice Registers	54	0	0	269200	0.02
Register as Flip Flop	54	0	0	269200	0.02
Register as Latch	0	0	0	269200	0.00
F7 Muxes	0	0	0	67300	0.00
F8 Muxes	0	0	0	33650	0.00

Figure 8: report_utilization

3.2 Simulazioni

Sono stati svolti svariati test su questo modulo. I test sono stati svolti sia in Behavioral che in Post-Synthesis Functional. Unica differenza che si può constatare è che in Post-Synthesis Functional vi sono dei ritardi nei segnali dovuti proprio alla sintesi del modulo. Questi ritardi, comunque, non causano problemi nei test. Ora verranno elencati i test svolti :

Test con sequenze di utilizzo: Si è andati a testare la possibilità di utilizzare il modulo per X volte di seguito.

Test con 0 bit di indirizzo: Si è testato se il modulo gestisse il numero minimo di bit indirizzo.

Test con 16 bit di indirizzo: Si è testato se il modulo gestisse il numero massimo di bit indirizzo.

Test reset sincrono: Nonostante il reset sia asincrono si è andati a testare questa eventualità per gestire il caso in cui il reset avvenisse durante il fronte di salita del clock.

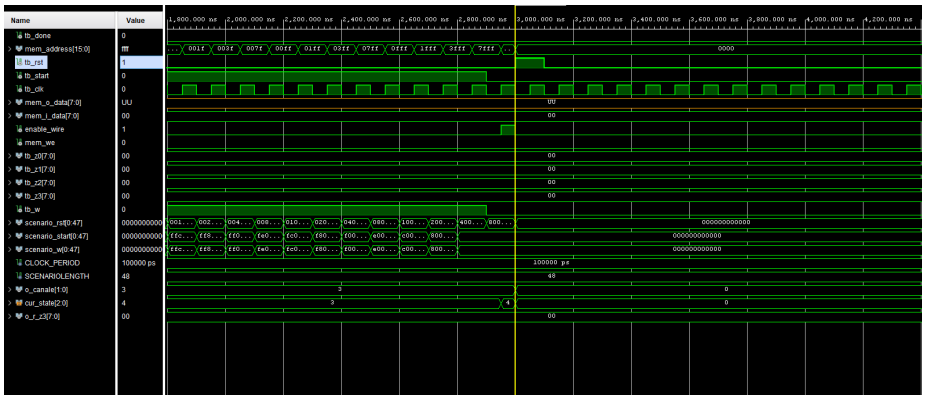

Figure 9: reset sincrono

Test reset asincrono durante o_done alto: Si è andati a testare se venisse resettato il modulo nel caso in cui arrivasse il reset alto durante done alto.

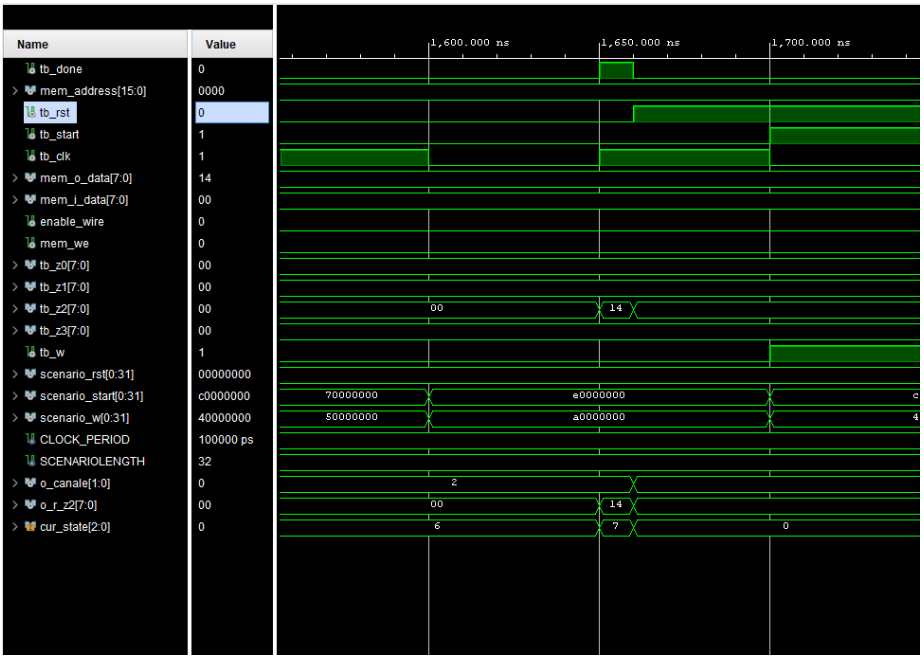

Figure 10: reset asincrono durante o_done alto

Test reset asincrono durante i_start alto: Si è andati a testare se venisse gestito correttamente un reset durante il periodo in cui i_start fosse alto. Come si può notare non appena il reset si alza lo stato ritorna in S0 e non viene considerato lo i_start alto. Questo lo si capisce dal canale di uscita che al posto di essere Z2 è Z1.

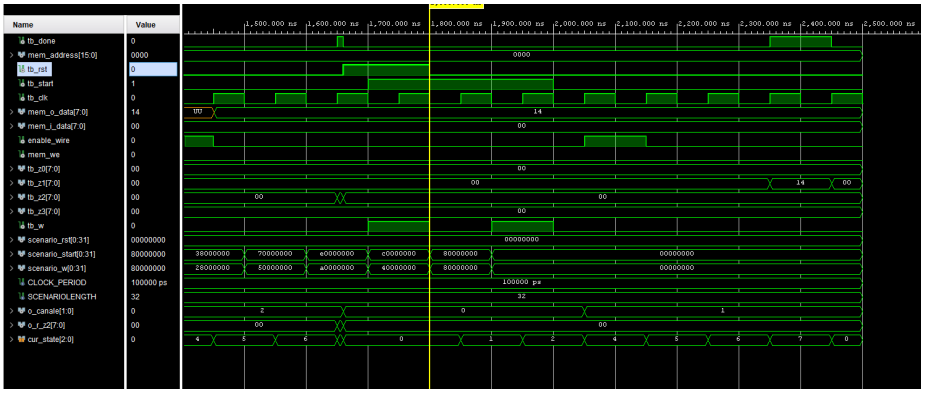

Figure 11: reset asincrono durante i_start alto

Test verifica sui valori dei segnali alla partenza: Si è andati a verificare che all'accensione vengano correttamente assegnati dei valori ai segnali. Nello specifico per i segnali riferiti alla memoria si è andati a verificare che non fossero U dato che sarebbe un errore considerarli tali.

Test di verifica sui fronti di salita e discesa: Si sono svolti dei test per verificare il corretto funzionamento durante i fronti di salita e discesa dei segnali.

Tutti questi test sono stati passati correttamente.

4 Conclusioni

Dati i risultati dei test si può concludere che il progetto svolge le funzionalità richieste nei tempi massimi richiesti(100 ns di ciclo di clock). Il progetto non contiene Latch, ma solo Flip-Flop. Il codice VHDL si può facilmente ricondurre al datapath e alla macchina a stati, cosa che facilità di molto la lettura e le future aggiunte/richieste. Il progetto è stato svolto utilizzando Artix-7 FPGA xc7a200tfbg484-1 ed il linguaggio VHDL.

5 Appendice-SCHEMATIC

Qui di seguito gli schemi del progetto acquisiti da Vivado.

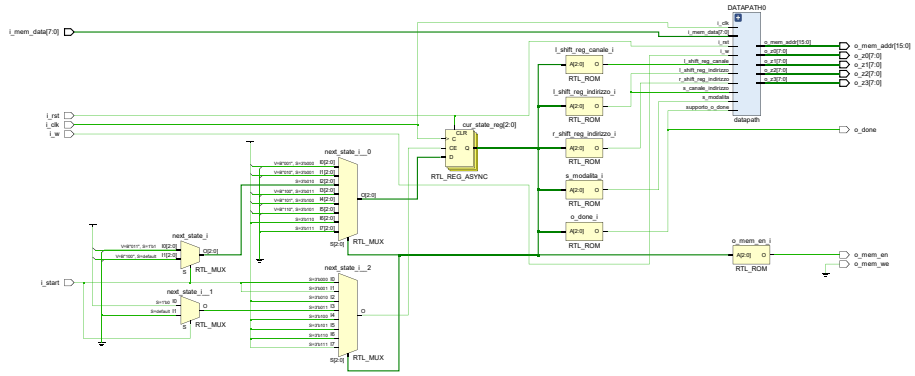

Figure 12: RTL ANALYSIS-1

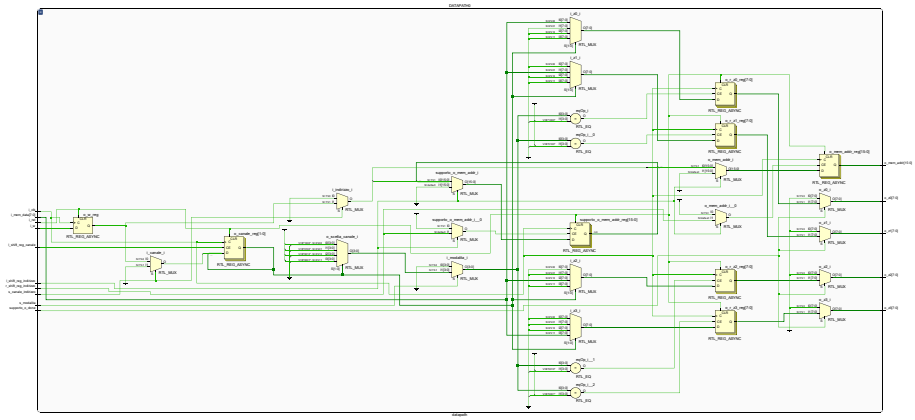

Figure 13: RTL ANALYSIS-2

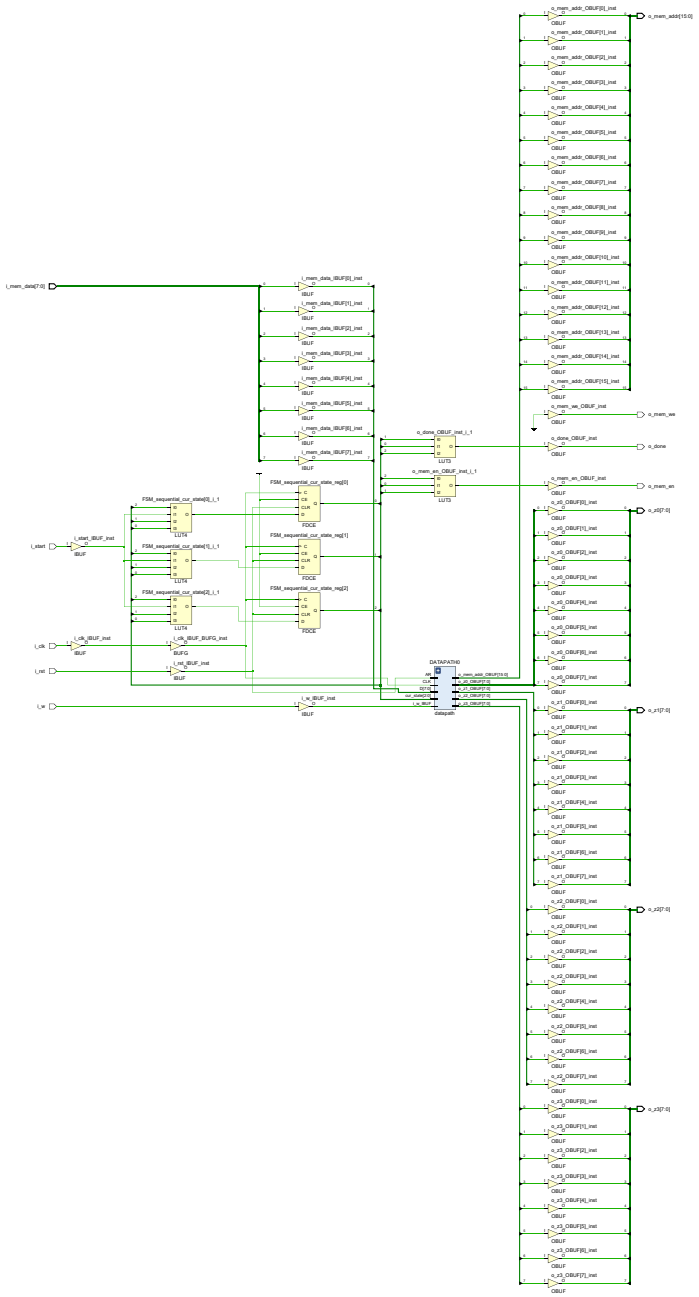

Figure 14: SYNTHESIS-1

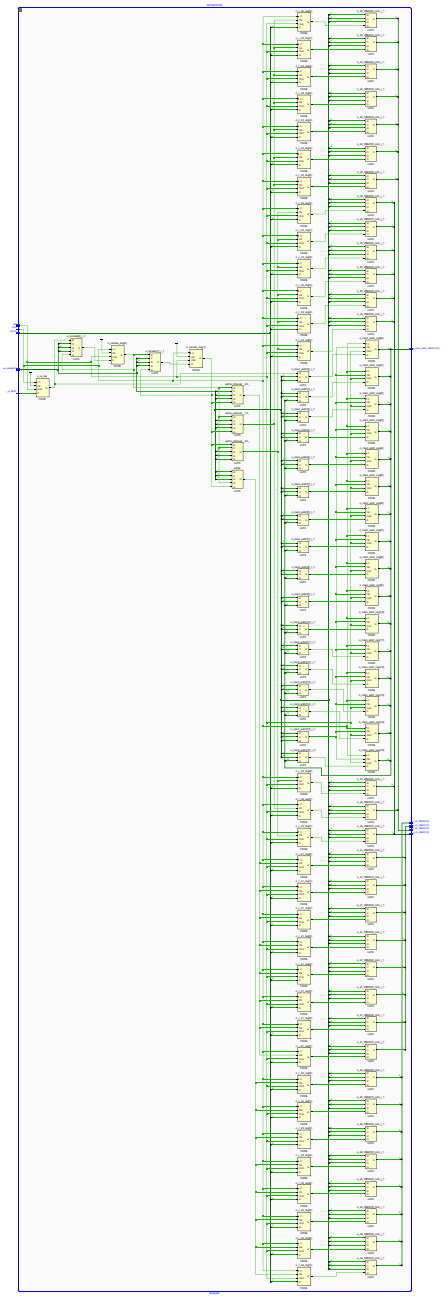

Figure 15: SYNTHESIS-2