CP: 10680459 Lorenzo Luisi

RELAZIONE PROVA FINALE RETI LOGICHE

INTRODUZIONE

Il progetto di reti logiche 2022/2023 consiste nell'implementare un circuito con quattro porte di ingresso: *i_clk*, *i_rst*, *i_start* e *i_w*. Il dispositivo fornisce un output in parallelo su una delle quattro uscite disponibili: *o_z0, o_z1, o_z2, o_z3*. È presente una quinta uscita *o_done* la quale notifica che l'elaborazione degli ingressi è terminata e il risultato viene trasmesso in output.

Gli ingressi *i_clk* e *i_rst* sono rispettivamente il segnale di clock e di reset unici per tutto il sistema, mentre *i_start* e *i_w* servono per fornire i dati in ingresso. Il segnale *i_rst* viene portato sul fronte alto all'accensione del dispositivo, e nel caso si voglia tornare allo stato iniziale. *i_w* è il segnale seriale che trasporta l'informazione in entrata, mentre *i_start* , quando sul fronte alto, indica che i dati in ingresso su *i_w* sono validi; altrimenti i dati in entrata su *i_w* non sono da considerare per il funzionamento del dispositivo.

Quando il segnale *i_start* è sul fronte alto, i primi due bit di *i_w* forniscono l'informazione relativa all'uscita su cui trasmettere il risultato. I restanti bit di *i_w*, al massimo sedici, indicano l'indirizzo della memoria esterna, nel quale è immagazzinato il risultato da trasmettere sull'uscita trovata precedentemente. Nel caso i bit validi successivi ai primi due siano meno di sedici, a questi verrà eseguito un padding di '0' a sinistra, per conformarli alla lunghezza degli indirizzi della memoria esterna.

Dopo aver eseguito il padding, se necessario, si esegue una lettura su memoria, che restituirà 8 bit di dati da trasmette sul canale corretto di uscita del dispositivo.

Le quattro uscite *o_z0, o_z1, o_z2, o_z3*, manterranno in memoria l'ultimo risultato che verrà trasmesso attraverso il loro canale; questo risultato verrà mostrato unicamente quando il segnale *o_done* si trova sul suo fronte alto per un singolo ciclo di clock, altrimenti i canali di uscita trasmetteranno la stringa di default '00000000'.

Esempio di funzionamento:

input:

i_rst: 10000000000000000000000

i_start: 00000011111111110000000

i_w: 01000100101010010011101

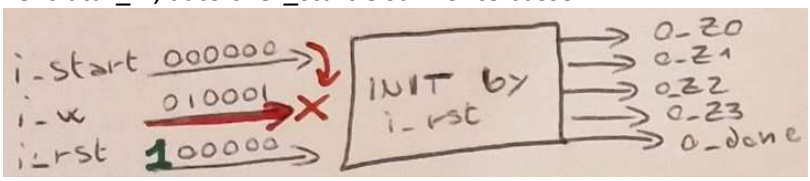
uscita: o_z0

indirizzo: 0000000010101001

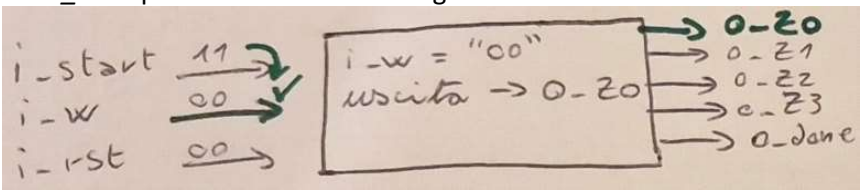
RAM all'indirizzo '0000000010101001': '10100111'

i bit rossi indicano quando *i_start* = '0' e quindi *i_w* trasporta dati non utili ai fini del funzionamento del dispositivo. I primi due bit verdi sono quelli relativi al canale di uscita, in questo caso '00' indica l'uscita o_z0, mentre la stringa gialla è l'indirizzo di memoria che dovrà subire un padding a sinistra per arrivare alla lunghezza di 16 bit .

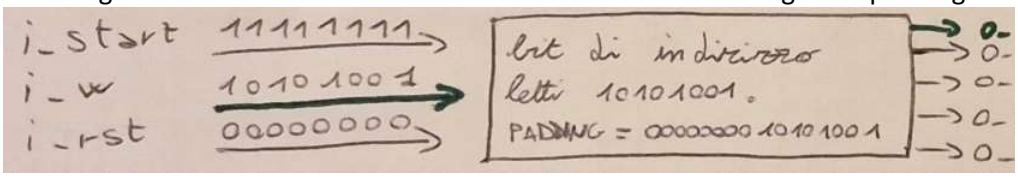
1. All'inizio il dispositivo viene inizializzato dal segnale di reset e non considera l'entrata *i_w*, dato che *i_start* è sul fronte basso.



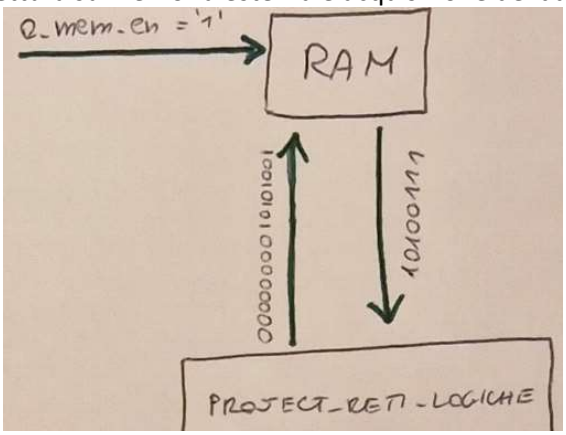
2. Appena *i_start* passa sul fronte alto vengono letti i bit relativi all'uscita corretta



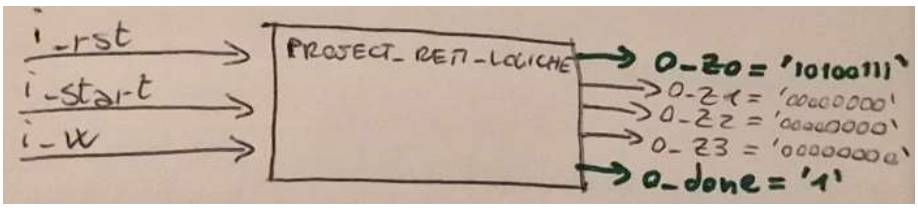
3. Vengono letti i bit relativi all'indirizzo di memoria e viene eseguito il padding



4. Lettura su memoria esterna e acquisizione dei dati

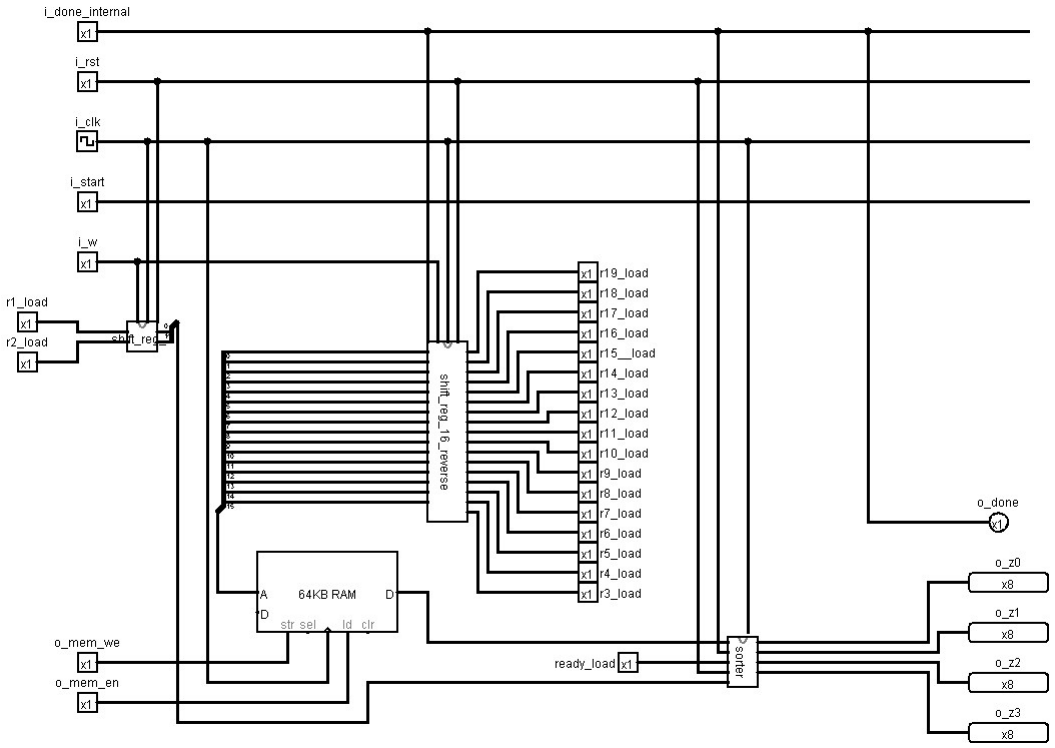


5. *o_done* viene portato a '1' e le uscite mostrano i dati aggiornati



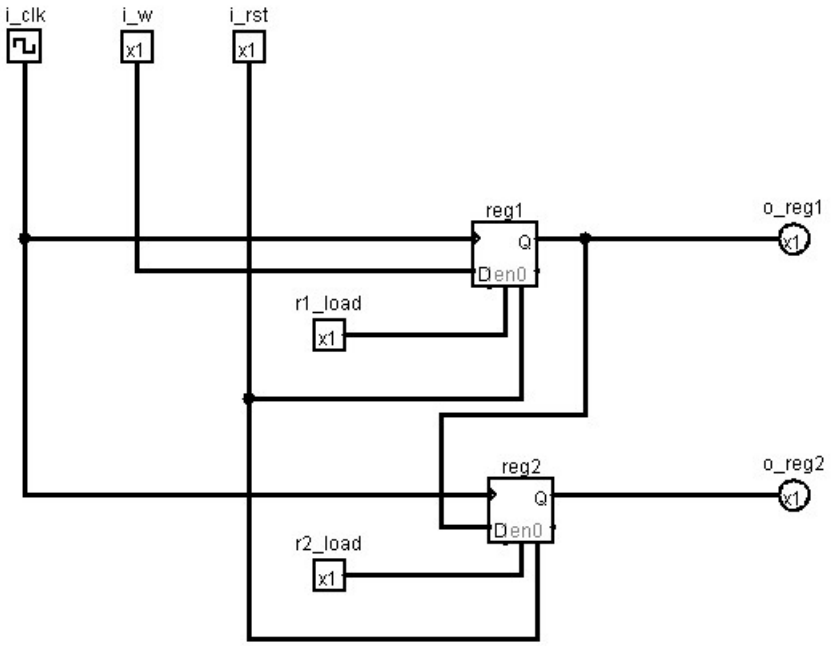
ARCHITETTURA

circuito complessivo con memoria esterna rappresentata:



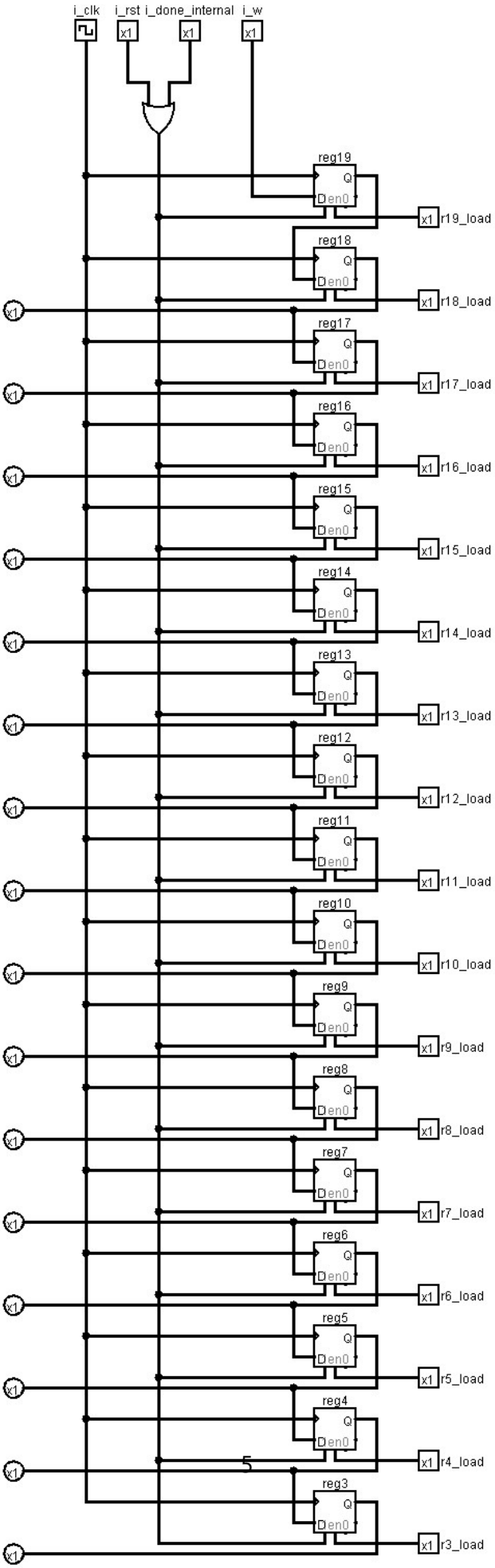
Modulo 1: shift_reg_sp2

Il primo modulo del circuito è composto da due process che descrivono ciascuno il comportamento di un registro. Essi vengono utilizzati per acquisire i primi due bit relativi al canale di uscita della sequenza corrente. Il primo prende i dati dall'ingresso da *i_w* e ha la sua uscita collegata al secondo registo, implementando così uno shift register a 2 bit. L'uscita del primo registro corrisponde al bit in posizione '0' dell'output del modulo, mentre l'uscita del secondo registro corrisponde al bit in posizione '1' dell'output del modulo, effettuando così un 'ribaltamento' dei segnali. I registri in questione vengono portati al valore di default '0' quando il segnale *i_rst* si trova sul fronte alto, in modo da essere reinizializzati. Come verrà mostrato con l'utimo modulo, che implementa la macchina a stati, non c'è bisogno di portare questi registri al valore di default tra una sequenza valida e quella successiva, dato che i bit per l'indirizzamento del canale vengono sempre forniti, e quindi sovrascriveranno tutto ciò che è presente in quel momento all'interno dei registri.



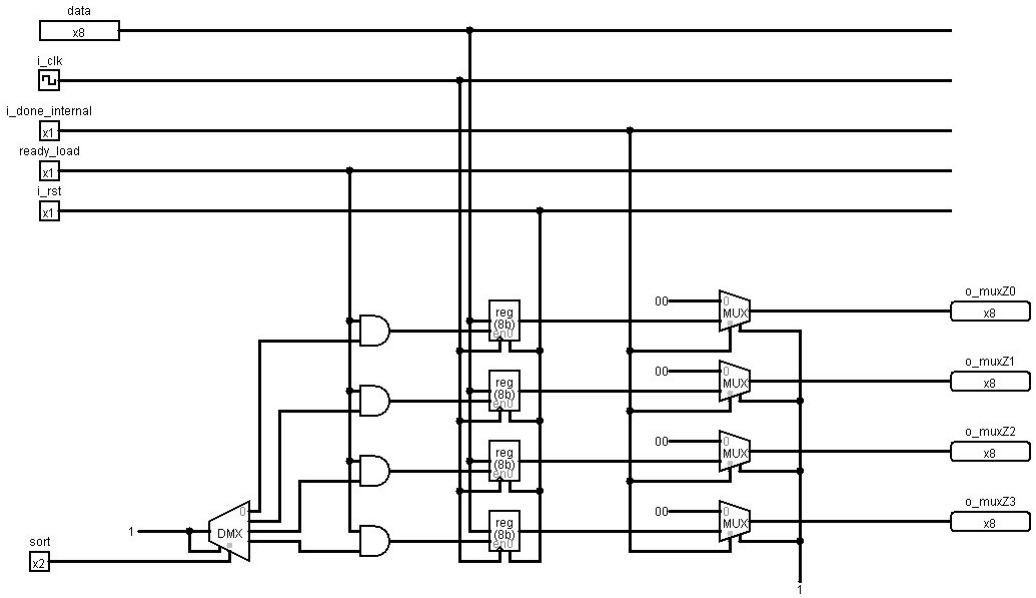
Modulo 2: shift_reg_sp16_reverse

Il secondo modulo è molto simile al primo, l'unica differenza è che vengono utilizzati 17 process per implementare dei registri, e di questi solo i primi 16 sono l'uscita effettiva che compongono l'indirizzo che andrà comunicato alla memoria esterna. Questi registri vengono portati tutti al valore di default tramite il segnale *i_rst* e il segnale *i_done_internal*, quest'ultimo alto quando il risultato finale è pronto per essere trasmesso; in questo modo, il modulo, è pronto a ricevere un eventuale nuova sequenza. I bit dell'indirizzo di memoria, come quelli relativi all'uscita, sono comunicati in modo seriale e quindi risultano ribaltati, ad esempio '110' sarebbe memorizzato nell'ordine '011', e dato che devono subire un padding di '0' per arrivare a una lunghezza di 16 bit il semplice 'ribaltamento' dei segnali in uscita non avrebbe portato all'output corretto. Quindi ho scelto di implementare uno shift register che acquisisca la sequenza trasmessa da *i_w* dal suo ultimo registro e poi propaghi con il passare dei cicli di clock i bit in ingresso ai restanti registri, in questo modo i bit di padding si trovano nella posizione corretta. Infine basta eseguire il consueto 'ribaltamento' dei segnali di uscita, ignorando l'ultimo registro, ottenendo in questo modo output del modulo. Per come è stata implementata la macchina a stati, e dato che l'informazione memorizzata in un registro è disponibile il ciclo dopo la lettura, ho deciso di aggiungere un registro in più che permettesse di far eseguire in modo corretto l'ultimo 'shift', così da non perdere un bit di informazione. Infatti se non avessi adottato questo approccio la macchina a stati, che gestisce tutti i moduli, avrebbe dovuto avere 15 stati aggiuntivi dopo lo stato S2 (vedi modulo 4), così facendo il costo in registri, relativi all'implementazione della macchina a stati, viene ridotto abbattendo anche degli ipotetici costi di produzione del dispositivo.


i_clk
i_rst
i_done_internal
i_w
reg19
reg18
reg17
reg16
reg15
reg14
reg13
reg12
reg11
reg10
reg9
reg8
reg7
reg6
reg5
reg4
reg3
r19_load
r18_load
r17_load
r16_load
r15_load
r14_load
r13_load
r12_load
r11_load
r10_load
r9_load
r8_load
r7_load
r6_load
r5_load
r4_load
r3_load

Modulo 3: sorter

Il terzo modulo gestisce l'imagazzinamento dell'output sul canale corretto e controlla i valori mostrati in uscita. Esso è composto da un insieme di process, che descrivono dei registri per la memorizzazione dei dati da mostrare in uscita dal dispositivo, inoltre vengono fornite le descrizioni dei quattro multiplexer collegati ciascuno all'output di un registro, e che utilizzano il segnale *i_done_internal* come selettore. In questo modulo prendo l'output del primo componente (*sort* in figura) e della memoria esterna (*data* in figura) per immagazzinare in modo corretto i dati. I bit uscenti dalla memoria vengono caricati su uno dei quattro registri, basandosi sul risultato del primo modulo, ad esempio se *sort* fosse '10' i dati della memoria saranno registrati sul registro di *o_z2*. Infine i multiplexer se *i_done_internal* si trova sul fronte alto, mostrano alle uscite *o_z0, o_z1, o_z2* e *o_z3* il valore dei registri corrispondenti, altrimenti mostrano '00000000'.



NOTA: il demultiplexer in figura serve a mostrare una possibile implementazione post sintesi, che può essere utilizzata per eseguire la condizione:
if(sort = "XX" AND ready_store = '1')
presente in ciascun registro per gestire correttamente l'imagazzinamento dei dati forniti dalla memoria esterna.

Modulo 4: macchina di Mealy

L'ultimo modulo del progetto è un insieme di process che descrivono il funzionamento di una macchina a stati di Mealy, utilizzata per manipolare i segnali interni, in modo da garantire il corretto funzionamento del dispositivo. Inoltre, qui viene fatto il collegamento tra il *segnale i_done_internal* e la porta di uscita *o_done* del dispositivo. La macchina a stati proposta è composta da sei stati: S0, S1, S2, S3, S4, S5.

S0 è lo stato iniziale della macchina, dove si ritorna nel caso il segnale *i_rst* sia sul suo fronte alto o se si ha terminato la computazione di una stringa di bit. Inoltre qui viene posto *r1_load* sul suo fronte alto, in modo da consentire la lettura del primo bit di indirizzamento

dell'uscita.

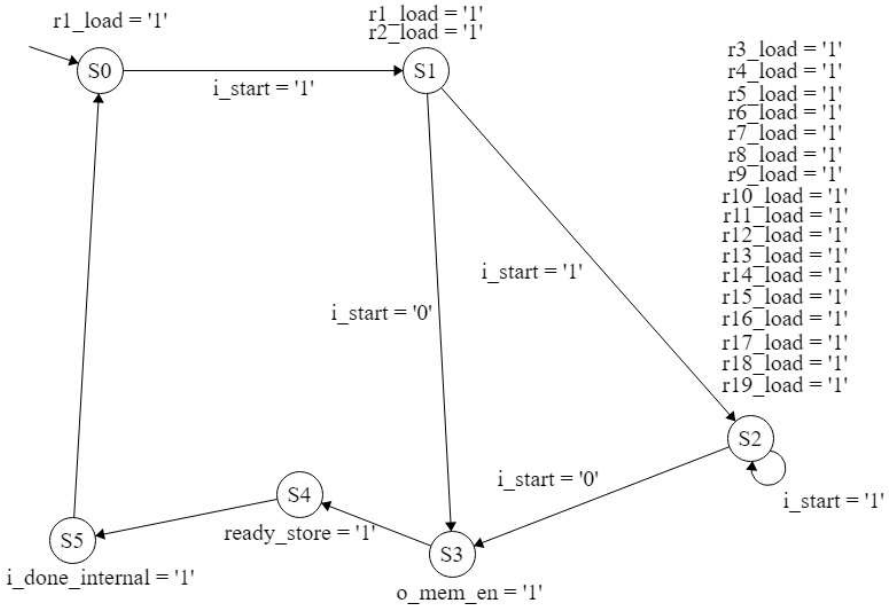
Dallo stato S0 si passa allo stato S1, al ciclo di clock successivo, nel caso il segnale *i_start* sia sul suo fronte alto. Qui troviamo *r1_load* e *r2_load* alti, in modo da acquisire anche il secondo bit di indirizzamento del canale di uscita. Da S1 si passa ad S2 nel caso *i_start* sia ancora sul suo fronte alto, mentre si passa a S3 nel caso sia sul fronte basso

In S2 abbiamo tutti i segnali da *r3_load* a *r19_load* alti, in modo da iniziare a memorizzare i dati relativi all'indirizzo da leggere nella memoria. Si rimane in questo stato per tutti i cicli di clock in cui il segnale *i_start* = '1', mentre si passa allo stato S3 solo quando *i_start* passa al fronte basso.

In S3 si porta il segnale *o_mem_en* = '1', in modo da poter leggere il contenuto della memoria all'indirizzo da 16 bit fornito dal secondo modulo. Da S3 poi si passa allo stato S4.

In S4 si porta il segnale *ready_store* = '1', in modo da immagazzinare sul registro corretto delle uscite il contenuto che ci viene fornito dalla memoria esterna, nello stato S3. Come visto nell'immagine relativa al modulo 3, la combinazione di *ready_store* e del segnale *sort,* permettono di scegliere il registro corretto su cui immagazzinare l'output della memoria esterna.

In S5 invece si porta *i_done_internal*='1', in modo da comunicare che l'elaborazione del dispositivo è terminata, e per mostrare in uscita il contenuto dei registi portando il selettore di tutti i multiplexer a '1'.



NOTA: nelle immagini rappresentative del circuito, non viene mai rappresentata l'operazione di 'ribaltamento' dei segnali del modulo 1 e del modulo 2, per questioni di ordine del disegno.

RISULTATI SPERIMENTALI

TIMING REPORT

Slack (MET) : 97.007ns (required time - arrival time)

Source: FSM_sequential_cur_state_reg[2]/C

(rising edge-triggered cell FDCE clocked by clock {rise@0.000ns fall@50.000ns period= 100.000ns})

Destination: SHIFT_REG_DATA/o_reg10_reg/CLR

(recovery check against rising-edge clock clock {rise@0.000ns fall@50.000ns period= 100.000ns})

Path Group: **async_default**

Path Type: Recovery (Max at Slow Process Corner)

Requirement: 100.000ns (clock rise@100.000ns - clock rise@0.000ns)

Data Path Delay: 2.404ns (logic 0.751ns (31.240%) route 1.653ns (68.760%))

Logic Levels: 1 (LUT4=1)

Clock Path Skew: -0.145ns (DCD - SCD + CPR)

Destination Clock Delay (DCD): 2.100ns = (102.100 - 100.000)

Source Clock Delay (SCD): 2.424ns

Clock Pessimism Removal (CPR): 0.178ns

Clock Uncertainty: 0.035ns ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE

Total System Jitter (TSJ): 0.071ns

Total Input Jitter (TIJ): 0.000ns

Discrete Jitter (DJ): 0.000ns

Phase Error (PE): 0.000ns

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[2]/C

FDCE (Prop_fdce_C_Q) 0.456 2.880 f FSM_sequential_cur_state_reg[2]/Q

net (fo=45, unplaced) 0.821 3.701 SHIFT_REG_DATA/cur_state[2]

LUT4 (Prop_lut4_I1_O) 0.295 3.996 f SHIFT_REG_DATA/o_reg19_i_2/O

net (fo=17, unplaced) 0.832 4.828 SHIFT_REG_DATA/o_reg19_i_2_n_0

FDCE f SHIFT_REG_DATA/o_reg10_reg/CLR

(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 SHIFT_REG_DATA/CLK

FDCE r SHIFT_REG_DATA/o_reg10_reg/C

clock pessimism 0.178 102.279

clock uncertainty -0.035 102.243

FDCE (Recov_fdce_C_CLR) -0.409 101.834 SHIFT_REG_DATA/o_reg10_reg

---

required time 101.834

arrival time -4.828

---

slack 97.007

UTILIZATION REPORT

1. Slice Logic

+-------------------------+------+-------+------------+-----------+-------+

| Site Type | Used | Fixed | Prohibited | Available | Util% |

+-------------------------+------+-------+------------+-----------+-------+

| Slice LUTs* | 27 | 0 | 0 | 134600 | 0.02 |

| LUT as Logic | 27 | 0 | 0 | 134600 | 0.02 |

| 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 |

+-------------------------+------+-------+------------+-----------+-------+

1.1 Summary of Registers by Type

+-------+--------------+-------------+--------------+

| Total | Clock Enable | Synchronous | Asynchronous |

+-------+--------------+-------------+--------------+

| 0 | _ | - | - |

| 0 | _ | - | Set |

| 0 | _ | - | Reset |

| 0 | _ | Set | - |

| 0 | _ | Reset | - |

| 0 | Yes | - | - |

| 0 | Yes | - | Set |

| 54 | Yes | - | Reset |

| 0 | Yes | Set | - |

| 0 | Yes | Reset | - |

+-------+--------------+-------------+--------------+

2. Memory

+----------------+------+-------+------------+-----------+-------+

| Site Type | Used | Fixed | Prohibited | Available | Util% |

+----------------+------+-------+------------+-----------+-------+

| Block RAM Tile | 0 | 0 | 0 | 365 | 0.00 |

| RAMB36/FIFO* | 0 | 0 | 0 | 365 | 0.00 |

| RAMB18 | 0 | 0 | 0 | 730 | 0.00 |

+----------------+------+-------+------------+-----------+-------+

3. DSP

+-----------+------+-------+------------+-----------+-------+

| Site Type | Used | Fixed | Prohibited | Available | Util% |

+-----------+------+-------+------------+-----------+-------+

| DSPs | 0 | 0 | 0 | 740 | 0.00 |

+-----------+------+-------+------------+-----------+-------+

4. IO and GT Specific

+-----------------------------+------+-------+------------+-----------+-------+

| Site Type | Used | Fixed | Prohibited | Available | Util% |

+-----------------------------+------+-------+------------+-----------+-------+

| Bonded IOB | 63 | 0 | 0 | 285 | 22.11 |

| Bonded IPADs | 0 | 0 | 0 | 14 | 0.00 |

| Bonded OPADs | 0 | 0 | 0 | 8 | 0.00 |

| PHY_CONTROL | 0 | 0 | 0 | 10 | 0.00 |

| PHASER_REF | 0 | 0 | 0 | 10 | 0.00 |

| OUT_FIFO | 0 | 0 | 0 | 40 | 0.00 |

| IN_FIFO | 0 | 0 | 0 | 40 | 0.00 |

| IDELAYCTRL | 0 | 0 | 0 | 10 | 0.00 |

| IBUFDS | 0 | 0 | 0 | 274 | 0.00 |

| GTPE2_CHANNEL | 0 | 0 | 0 | 4 | 0.00 |

| PHASER_OUT/PHASER_OUT_PHY | 0 | 0 | 0 | 40 | 0.00 |

| PHASER_IN/PHASER_IN_PHY | 0 | 0 | 0 | 40 | 0.00 |

| IDELAYE2/IDELAYE2_FINEDELAY | 0 | 0 | 0 | 500 | 0.00 |

| IBUFDS_GTE2 | 0 | 0 | 0 | 2 | 0.00 |

| ILOGIC | 0 | 0 | 0 | 285 | 0.00 |

| OLOGIC | 0 | 0 | 0 | 285 | 0.00 |

5. Clocking

| Site Type | Used | Fixed | Prohibited | Available | Util% |

| BUFGCTRL | 1 | 0 | 0 | 32 | 3.13 |

| BUFIO | 0 | 0 | 0 | 40 | 0.00 |

| MMCME2_ADV | 0 | 0 | 0 | 10 | 0.00 |

| PLLE2_ADV | 0 | 0 | 0 | 10 | 0.00 |

| BUFMRCE | 0 | 0 | 0 | 20 | 0.00 |

| BUFHCE | 0 | 0 | 0 | 120 | 0.00 |

| BUFR | 0 | 0 | 0 | 40 | 0.00 |

+------------+------+-------+------------+-----------+-------+

6. Specific Feature

+-------------+------+-------+------------+-----------+-------+

| Site Type | Used | Fixed | Prohibited | Available | Util% |

+-------------+------+-------+------------+-----------+-------+

| BSCANE2 | 0 | 0 | 0 | 4 | 0.00 |

| CAPTUREE2 | 0 | 0 | 0 | 1 | 0.00 |

| DNA_PORT | 0 | 0 | 0 | 1 | 0.00 |

| EFUSE_USR | 0 | 0 | 0 | 1 | 0.00 |

| FRAME_ECCE2 | 0 | 0 | 0 | 1 | 0.00 |

| ICAPE2 | 0 | 0 | 0 | 2 | 0.00 |

| PCIE_2_1 | 0 | 0 | 0 | 1 | 0.00 |

| STARTUPE2 | 0 | 0 | 0 | 1 | 0.00 |

| XADC | 0 | 0 | 0 | 1 | 0.00 |

+-------------+------+-------+------------+-----------+-------+

7. Primitives

+----------+------+---------------------+

| Ref Name | Used | Functional Category |

+----------+------+---------------------+

| FDCE | 54 | Flop & Latch |

| OBUF | 51 | IO |

| LUT4 | 36 | LUT |

| IBUF | 12 | IO |

| LUT5 | 4 | LUT |

| LUT3 | 4 | LUT |

| LUT2 | 1 | LUT |

| BUFG | 1 | Clock |

+----------+------+---------------------+

SIMULAZIONI

In questa sezione riporto i test bench che sono stati effettuati in modo autonomo, dopo aver verificato che ciascun test fornito dai docenti fosse superato correttamente.

Test bench I: In questo test bench, si controlla che il dispositivo funzioni con qualsiasi lunghezza del messaggio, relativo all'indirizzo di memoria (da 0 fino a 16 bit). Osservando che in ogni caso il segnale *i_start,* sia sul suo fronte alto per un minimo di due cicli di clock e un massimo di diciotto cicli di clock. Altrimenti si andrebbe in contro a una violazione della specifica del dispositivo, quindi il corretto funzionamento non può essere garantito. Inoltre, il test si assicura che le uscite siano aggiornate correttamente tra una stringa valida e quella successiva. Viene anche controllato che il segnale di reset riporti il dispositivo alla configurazione iniziale e che non utilizzi mai più di 20 cicli di clock, dopo che *i_start* passa al fronte basso, per completare la computazione dei dati.

input:

i_rst:
01000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000010000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000010000000000000000000000000000000000000
00000000000000000000000000000000

i_start:
00001100000000000000000000111000000000000000000000111100000000000000000000
11111000000000000000000011111100000000000000000000111111100000000000000000
00000111111111000000000000000000001111111110000000000000000000011111111110
00000000000000000001111111111100000000000000000000111111111111000000000000
00000000111111111111100000000000000000001111111111111110000000000000000000
00011111111111111100000000000000000011111111111111111000000000000000000001
11111111111111111000000000000000000011111111111111111000000000000000000000
11111111111111111000000000000000

i_w:
00000000000000000000000001010000000000000000000011100000000000000000000000
11110000000000000000000011001000000000000000000001101100000000000000000000

00000101110010000000000000000000011000101000000000000000000000110101100
00000000000000000000111101001000000000000000000000011011010001000000000
00000000010101001000000000000000000000000011101111110011000000000000000
00010111100101010100000000000000000000010011010101010000000000000000000
11001101010010100000000000000000000010100110101001010000000000000000000
10111111111111111100000000000000000

lunghezza dello scenario: 549

Test bench II: il test controlla che l'output del dispositivo sia sempre '0' dato che il segnale *i_start* è sempre basso. In questo caso il segnale di reset viene sempre tenuto basso, ad eccezione per i primi 3 bit, in modo da verificare correttamente che il dispositivo non dia risultati differenti da 0 su una qualsiasi uscita

input:

i_rst:
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000

i_start:
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000

i_w:
00000100110000000000001100000000011111111111100000000000000100000000000
10000001110000000001000100000000000000000010000001000011000000100100100
01001010001010100000000000001010100101000010000010011

lunghezza dello scenario: 200

osservazione: Ho deciso di non eseguire un ultimo caso di testing, ovvero quello in cui sia *i_start* sia *i_rst* siano sul fronte alto contemporaneamente . In questo caso l'implementazione da me elaborata non fornirebbe un output corretto, dato che l'input non è conforme alle specifiche. Se si volesse gestire questo evento, si potrebbe aggiungere uno stato con tutti i segnali a '0' alla macchina di Mealy, dove si finisce nel caso si abbia la condizione:

"i_start = '1' AND i_rst = '1' "

dopodiché si rimarebbe in questo stato fino a quando i_start non torna sul suo fronte basso, facendo tornare la macchina a S0. Preciso che la sequenza corrente verrebbe così scartata. Tutti i registri da reg3 a reg19 e quelli relativi alle uscite, vengono reinizializzati dal fatto che i_rst fosse sul suo fronte alto nella condizione sopra citata.

CONCLUSIONI

In conclusione il progetto 2022/2023 di reti logiche, ha portato alla realizzazione di un dispositivo VHDL capace di ricevere dei dati in ingresso, ed elaborarli interfacciandosi con una memoria esterna per poi instradarli su una delle sue quattro uscite. Esso può essere adoperato come componente di un sistema più grande per la gestione interna dei dati, dove un secondo dispositivo comunicando in modo seriale, può instradare comandi o dati pre-memorizzati su una ROM, verso un terzo componente connesso a una delle uscite del dispositivo.

I risultati sperimentali mostrano che il dispositivo soddisfa le specifiche richieste, garantendo il funzionamento in una vasta gamma di scenari, come mostrato dai test bench, dove il circuito riesce a gestire qualsiasi lunghezza di input compresa tra i 2 e i 18 bit. Il report di timing, mostra che il dispositivo rispetta i requisiti temporali, con uno slack positivo significativo, indicando l'efficienza dal punto di vista temporale.

Vorrei ringraziare il professor Salice e il professor Terraneo per avermi permesso di crescere come studente e futuro professionista, dandomi l'opportunità di progettare in prima persona un circuito logico.

Lorenzo Luisi