Relazione del Progetto di Laboratorio
Architettura degli Elaboratori I

Edoardo Ferrari*
matricola: 892905, Turno: A
edoardo.ferrari2@studenti.unimi.it

1 Cifrario di Cesare

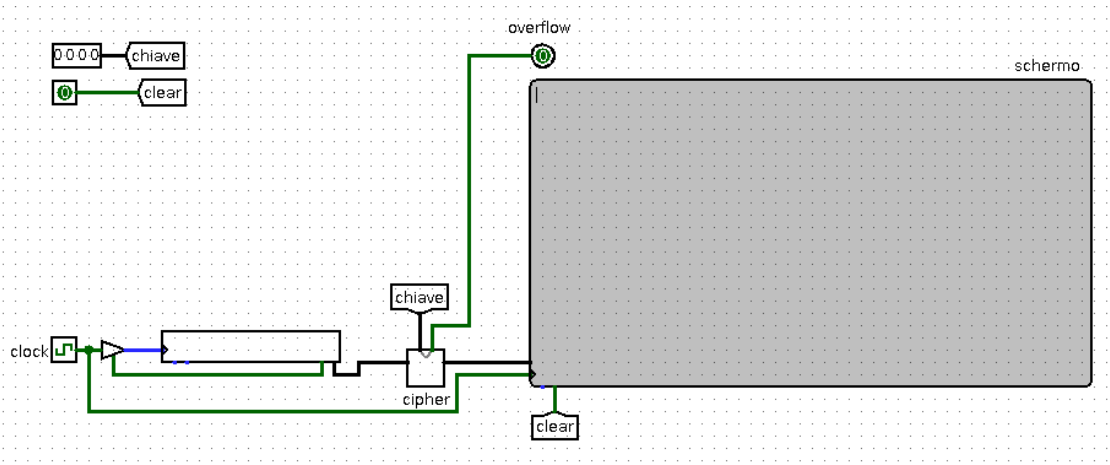

Figure 1: Screenshot dell'interfaccia principale

Il progetto consiste nel creare mediante Logisim un sistema in grado di cifrare un testo usando un Cifrario di Cesare. La chiave (di 4 bit massimi, sufficienti sia per simulare ROT3 originario sia ROT13, variante di ROT3) verrà immessa come input, mentre la stringa di testo verrà letta dalla tastiera. Il circuito ignorerà tutti i caratteri non alfabetici e si occuperà solamente di cifrare quelli alfabetici (della codifica ASCII classica), distinguendo tra maiuscole e minuscole. Il risultato verrà infine mostrato su un display.

1.1 Strumenti I/O

Di seguito gli strumenti utilizzati per interagire con il resto dei circuiti.

*Progetto approvato il 23/12/2019, consegnato il 10/01/2020

Nome	Scopo	Utilizzo
chiave	I	Scelta della chiave
clock	I	Se "enabled" scandisce il clock
clear	I	Se attivato, "pulisce" lo schermo
overflow	O	In caso di overflow, restituisce 1
schermo	O	Schermo per visualizzare il messaggio cifrato

2 Componenti realizzati

Per la realizzazione sono stati necessari 8 circuiti, 4 dei quali (HA, FA e i comparatori a 1 bit e 7 bit) sono stati realizzati per ridurre la complessità o per aiutarmi nella comprensione dei processi. Alcuni circuiti sono invece presi direttamente da Logisim in quanto la loro realizzazione non era strettamente inerente con la comprensione del progetto. Di seguito verranno quindi mostrati i fogli di lavoro grazie ai quali il progetto riesce a funzionare.

2.1 Cesare_sum

Questo circuito si occupa della somma tra il plaintext e la key. La chiave utilizza 4 bit perché, come già descritto all'inizio del documento, con 4 bit si possono rappresentare ROT3 e ROT13 (si può arrivare anche fino a ROT15).

Un'ulteriore motivazione per cui la chiave non è stata scelta di 5 bit o più, risiede nel fatto che non avrebbe senso permettere al circuito di svolgere ROT(N) con $N > 26$, essendo 26 le lettere dell'alfabeto. A tal proposito è utile creare un half adder: dato che si stanno sommando 7 bit con 4 bit, ci saranno 3 bit in cui non avverrà quasi nulla, basterà quindi rimandare il riporto e usarlo come input in un half adder, anziché implementare dei full adder. In questo modo posso risparmiare sul numero di porte logiche.

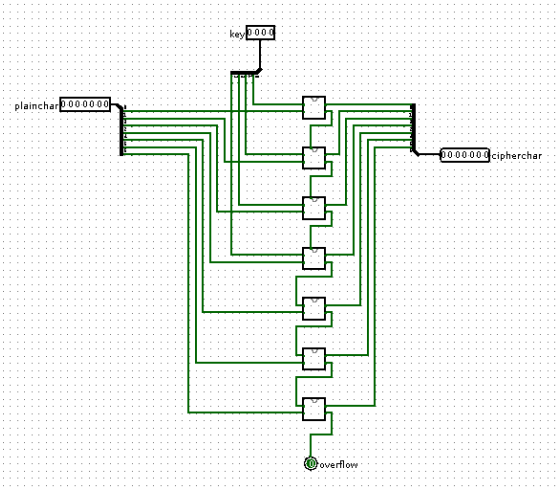

Figure 2: Screenshot di Cesare_sum

2.2 Cesare_op

Questo piano di lavoro è dedicato alla creazione del circuito che nel cifrario si occupa di ricodificare il valore ASCII e di compiere l'operazione modulo. Sono presenti costanti esadecimali come 'a'(61), 'A'(41) e '26'(1a) che permetteranno di svolgere i calcoli necessari. Sono inoltre presenti tre input, uno a 7 bit e due selettori rinominati "islower" e "isupper" che saranno utili nel circuito successivo. Si noti anche l'esistenza di un collegamento diretto tra l'input a 7 bit e il MUX, necessario nel caso in cui il valore ASCII non sia alfabetico. In ouput si avrà un altro valore a 7 bit.

Per capire meglio il funzionamento basti pensare che, come si vedrà nel circuito successivo, in ingresso si avrà il plainchar sommato alla key. Questo char dovrà essere ridotto di 'a' (o 'A' a seconda se minuscolo o maiuscolo), modulato, e il resto dovrà essere sommato sempre ad 'a' (o 'A'). In questo modo si ottiene il valore esatto del cipherchar in uscita.

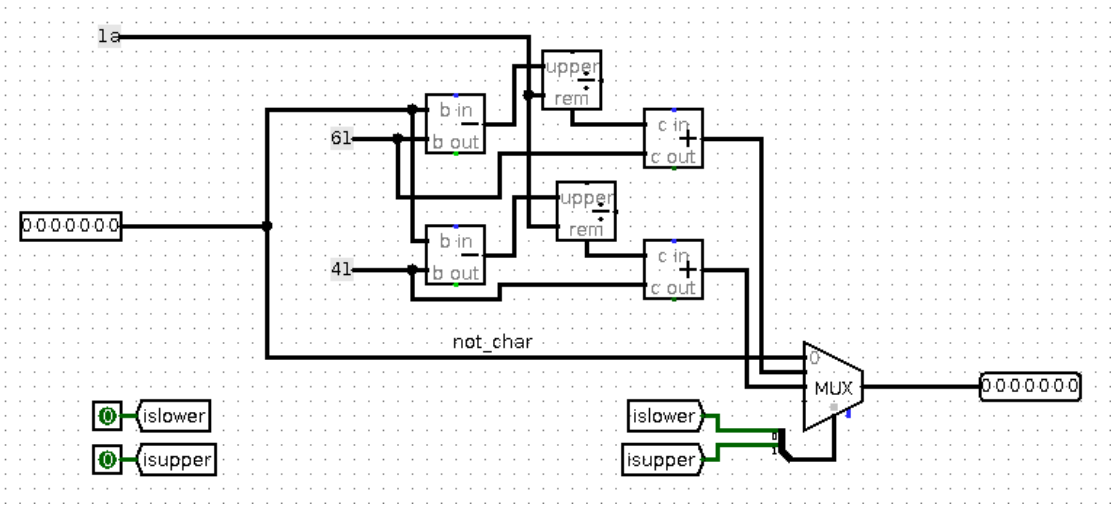

Figure 3: Screenshot di Cesare_op

2.3 Cesare

In questa sezione viene presentato il circuito che ha l'incarico di collegare tra loro tutti i componenti che permettono di svolgere l'operazione di cifratura. Gli input presenti sono appunto i 7 bit del carattere e i 4 bit della chiave che vengono immessi nel circuito Cesare_sum precedentemente illustrato. Sottostanti sono presenti i circuiti che permettono la distinzione tra maiuscole e miuscole: i valori immessi come costanti sono semplicemente i valori esadecimali delle lettere 'A-Z' e 'a-z'. Questi circuiti poi propagano il loro valore fino al circuito "mod", responsabile dell'operazione modulo e ricodifica.

Si noti che nel MUX presente al centro del quadrante, vengono immessi tre input, di cui uno è il dato originario del plainchar mentre gli altri sono la somma appena avvenuta tra plainchar e key. Questo perché nel caso in cui il valore del carattere non

sia compreso tra gli indici di "isupper" e "islower", significa che non è un carattere alfabetico e viene selezionato e ignorato nel circuito successivo, nel caso invece in cui lo fosse, viene selezionata la somma e essa viene ricodificata in "mod".

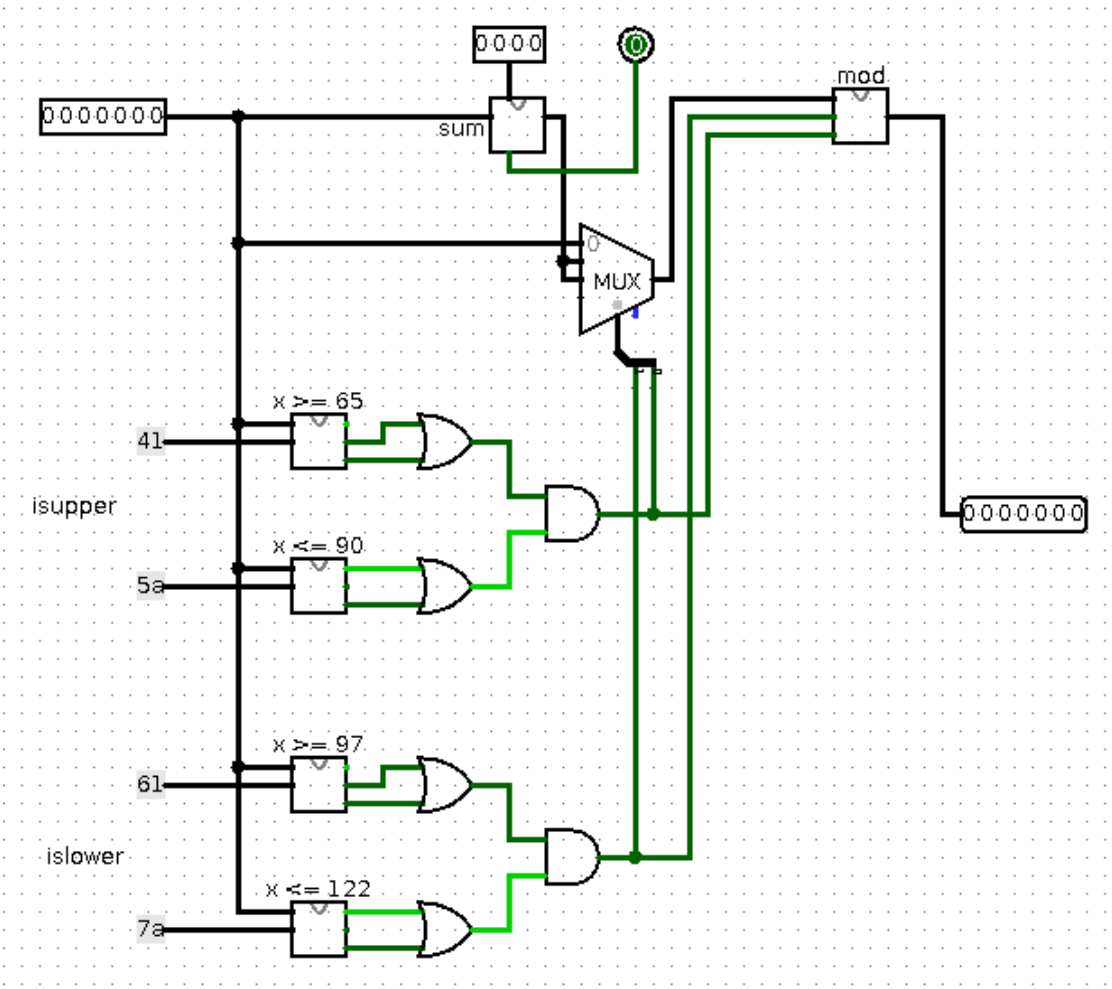

Figure 4: Screenshot di Cesare

3 Storico del processo

Inizialmente l'idea era quella di progettare solamente il circuito per implementare ROT3, ovvero il cifrario classico di Cesare, salvandomi tutti i valori delle lettere da 'A' a 'Z'. Tuttavia questo approccio poteva risultare uno spreco di porte logiche per un risultato praticamente minimo. Così ho deciso di progettare il circuito complessivo partendo dall'idea di un programma in C (indicativamente, qualsiasi linguaggio andrebbe bene): l'algoritmo infatti non ha una tabella con i valori prestabiliti ma si occupa di calcolare al

momento il ciphertext, risultando non solo più efficiente, ma anche più elastico. Grazie a questo approccio ho infatti potuto espandere il circuito: ho dato la possibilità all'utente di usare chiavi a scelta, entro un certo intervallo, e ho dato la possibilità di distinguere tra lettere maiuscole e minuscole, aumentando sì la complessità ma guadagnando un risultato vantaggioso ed elegante.