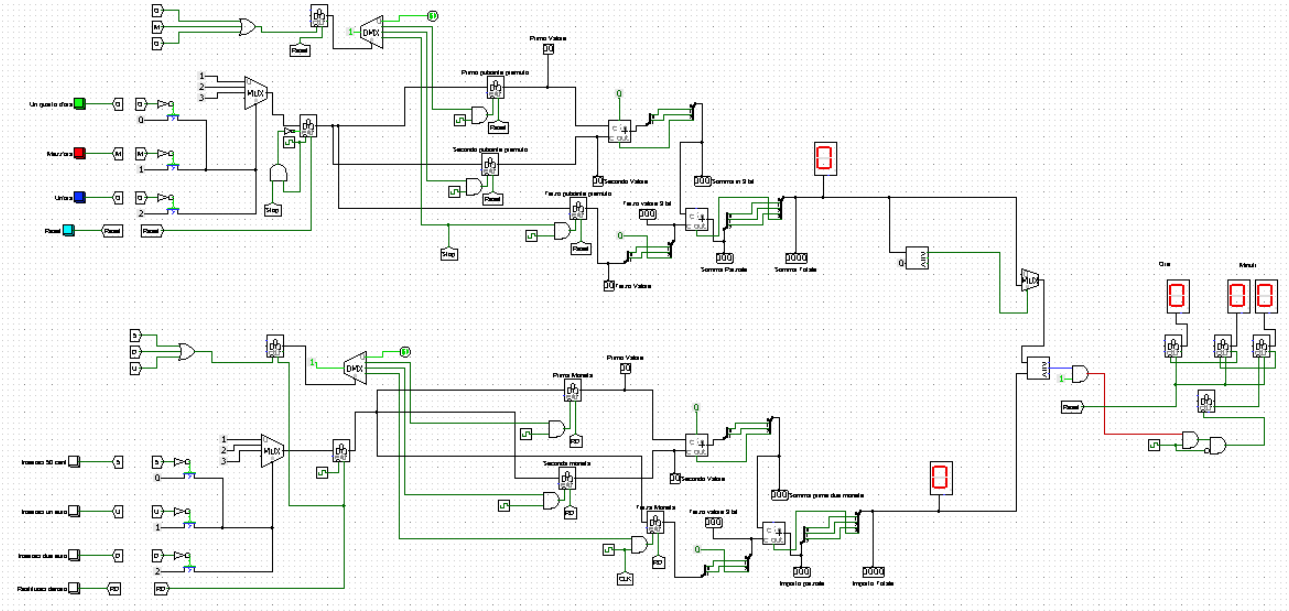
Relazione d’esame, Architettura degli Elaboratori I

Progetto in *Logisim*

Parchimetro



Specifiche del progetto

Funzionamento del parchimetro:

Il mio progetto consiste nella creazione di un parchimetro. Il circuito in figura è in grado di ricevere una quantità oraria (che varia da un minimo di 45 minuti ad un massimo di 3h) e, prima che scatti il conteggio dei minuti, viene richiesta una somma di denaro corrispondente al tempo richiesto. E’ possibile eliminare i dati inseriti, per quanto riguarda la fascia oraria, premendo il tasto “Reset”; è inoltre possibile chiedere una restituzione del denaro premendo il tasto “Restituisci Denaro” (in particolare questo tasto va ad annullare la somma inserita). Una volta introdotto correttamente l’importo, scatterà un “orologio” che indicherà quanto tempo è passato.

Canali di Input/Output:

Input: 3 bottoni per scegliere il range orario: uno corrispondente a 15 minuti, uno a 30 minuti e l'ultimo ad un'ora. 3 bottoni per introdurre il denaro; un bottone per 50 centesimi, il secondo per un euro e il terzo per due euro. Sono inoltre disponibili altri due tasti, "Reset" per la fascia oraria e "Restituisci Denaro" per il pagamento, che permettono di annullare le scelte.

Output: Due display esadecimali che permettono l'uno di vedere quanto dobbiamo pagare, l'altro quanto denaro abbiamo inserito. Infine un orologio formato da 3 display a esadecimali, di cui uno indica le "ore" passate, ed i restanti i "minuti" passati.

Stato iniziale della macchina:

La macchina attende che venga inserito il range orario (o l'importo, l'ordine è ininfluente) e attende che venga introdotta la somma di denaro corrispondente. Se non si preme prima dell'introduzione del denaro o il tasto Reset o il tasto Restituisci Denaro, la macchina entrerà in funzione.

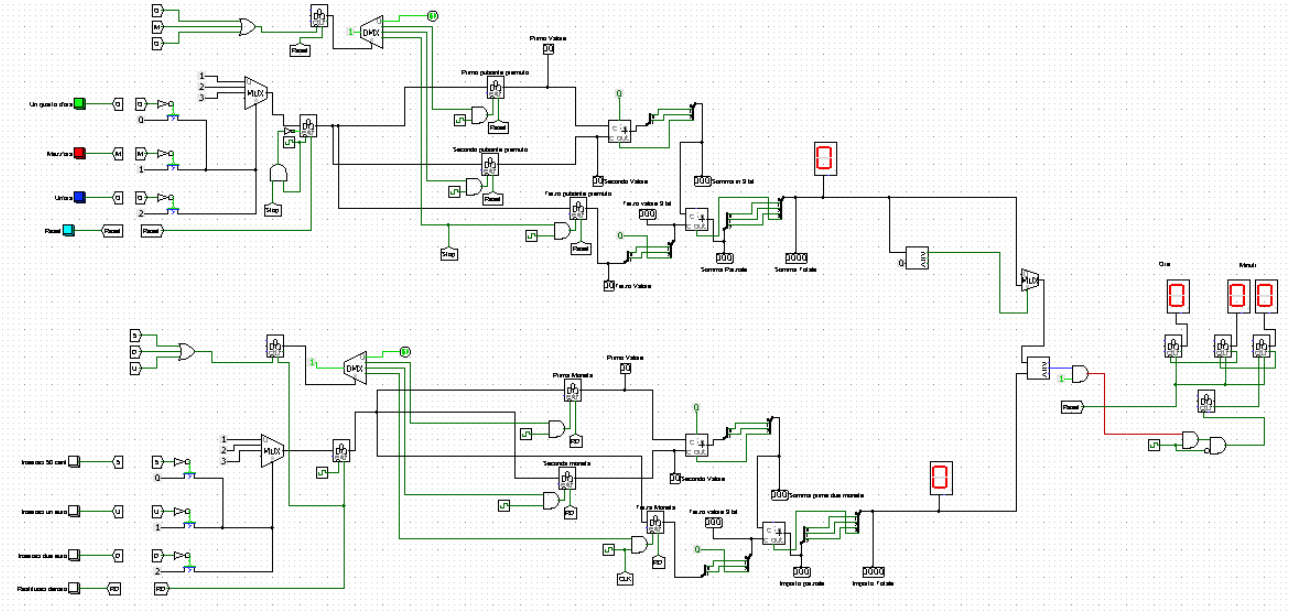
Esecuzione della macchina:

La macchina ha riconosciuto i valori da noi inseriti e, in caso di corretto inserimento dei dati, fa partire il tempo.

Stato Finale:

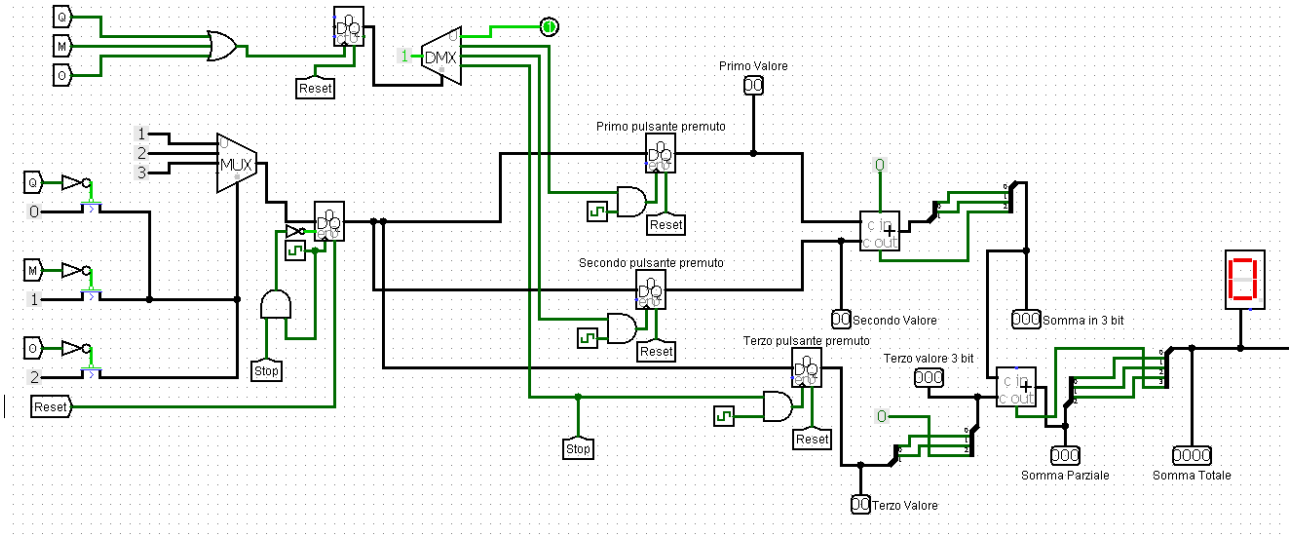
L'orologio entra in funzione ed è possibile vedere quanto tempo è passato. Se si vuol far eseguire un nuova istruzione alla macchina, è necessario prima fare un "Reset Simulation" dal tool di Logisim.

Circuito principale



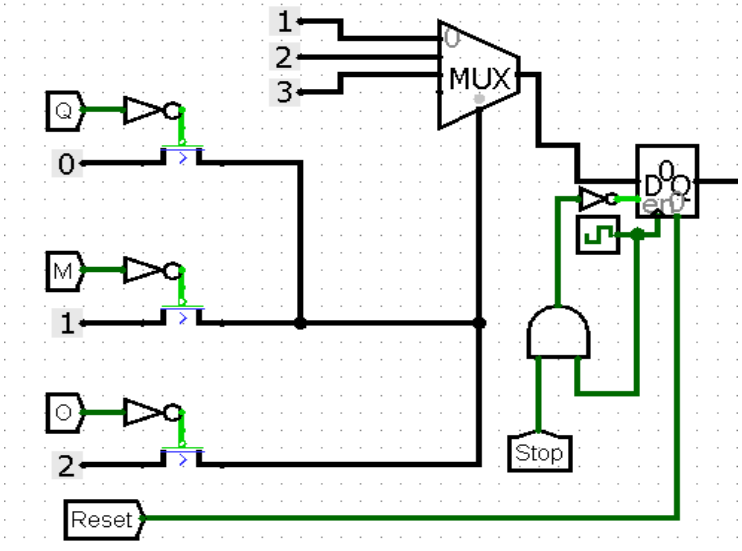
Questo è il vero e proprio circuito. Possiamo notare tutti i componenti citati nelle specifiche, come i pulsanti sulla sinistra, oppure i vari display sulla destra. Per fare il progetto ho subito cercato di pensare come un programmatore, ovvero secondo la logica DRY (Don’t Repeat Yourself), cercando dunque di riutilizzare quanto più circuito fosse possibile, sia per una facilità di progettazione, sia e soprattutto per poter fare velocemente modifiche al sistema su entrambi i fronti. Il circuito lo si può vedere come formato da tre grandi circuiti (per quanto due, come già detto, molto simili), il Blocco A, dedito alla fissazione di un range orario, il Blocco B, necessario per l’introduzione dell’importo corrispondente, ed il Blocco C, ovvero una comparazione ed un cronometro.

Blocco A



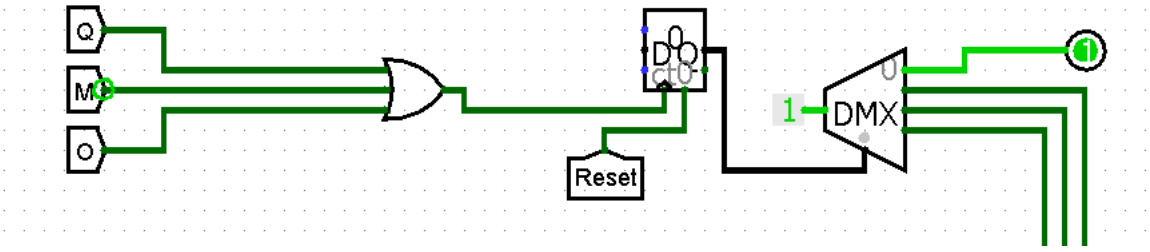
Funzionamento blocco A:

Una volta premuto uno dei rispettivi bottoni, il segnale arriva ad un mux che va a selezionare il pulsante premuto. Davanti a questa prima scelta circuitale, ho optato per l'utilizzo di 3 transistor in parallelo, transistor che hanno come segnale da lasciar passare o meno 0 per far passare il primo valore, 1 per far passare il secondo e 2 per far passare il terzo.



Come si vede dalla figura, ad ogni pulsante corrisponde un tunnel (Q – Quarto d'ora, M – Mezz'ora, O – Un'ora). Quando premo un pulsante, il segnale arriverà al transistor: il segnale tuttavia deve essere negato, perché altrimenti passerebbero tutti i segnali se non premessi nulla e due segnali quando premo un pulsante (in particolare i valori dei restanti due pulsanti). Il transistor funziona a due bit, perché per scelta circuitale ho imposto che si possano introdurre 3 dati, ed un singolo bit ovviamente non è in grado di coprire 3 valori diversi. Il segnale, una volta "lasciato passare" dal transistor, finisce nella locazione di selezione di un Multiplexer, a cui arrivano 3 segnali, corrispondenti ai 3 valori per i range di tempo (una quarta locazione rimane vuota). Questo segnale dunque come detto sopra andrà a selezionare il dato da noi richiesto, e lo proietterà in uscita. Il segnale, che andrà ora a rappresentare il valore del range orario, finirà in un registro a 2 bit. Arriva inoltre al registro un segnale alla locazione 0, questo è il Reset, necessario per eliminare i dati inseriti in caso di errore. Si nota anche un tunnel con scritto "Stop" negato, che vedremo dopo dove viene generato.

Il segnale dunque è nel register; mi chiedo dunque "cosa devo farne di questo valore"? Questo valore deve essere messo in un altro register, in modo che venga conservato: infatti quando io premo un altro bottone, il valore precedente viene dimenticato. Nasce un "problema": io ho 3 bottoni, di conseguenza avrò 3 valori, valori che ovviamente possono essere distinti; come posso dunque salvare un valore in un registro e "congelarlo" e poi salvare un altro valore nel registro successivo? La risposta che ho trovato consiste nell'introduzione di qualche tunnel, una porta OR, un counter a 2 bit e un Demultiplexer.

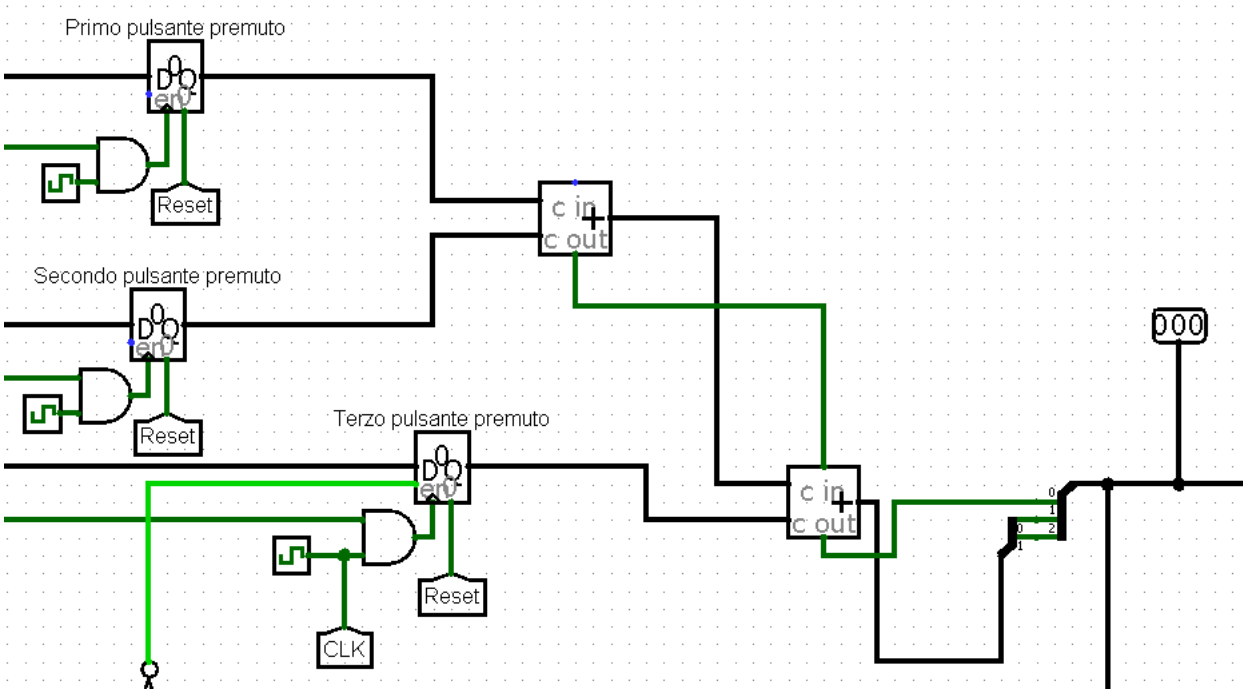


Osservando il circuito, si notano immediatamente dei tunnel; questi tunnel sono collegati ai bottoni. La porta è un OR per fare in modo che basti anche solo che io prema un bottone perché si asserisca la porta. Il segnale in uscita dall'OR finisce in un Counter (notare che nel counter alla locazione 0 è sempre presente Reset, tramite tunnel, per correggere sempre eventuali errori). Il segnale che arriva al counter finisce nella locazione del Clock, quindi questo

counter aumenterà il suo valore ogni volta che un bottone viene premuto. Il counter, che ha come valore massimo 3, è fatto in modo che si congeli quando raggiunge il massimo. Il segnale che esce dal counter finisce in un Demux, che riceve in entrata un valore 1, in modo che il componente sia sempre in funzione. In un primo momento avevo collegato il primo registro alla prima locazione, il secondo alla seconda e il terzo alla terza. Notai tuttavia che vi erano dei problemi, il valore veniva salvato infatti in registri errati. Osservando in particolare il demux capii dove era l'errore; il counter in un primo momento (ovvero quando non ho ancora premuto nessun bottone), invia il segnale 0 al demux, che va ad illuminare la corrispondente uscita, ovvero quella che io avevo pensato come "prima". Il problema dunque l'ho risolto agilmente andando a porre un Pin sulla prima uscita e collegando i registri che riceveranno i valori nelle uscite successive.

Dunque, dopo che il valore è passato nel primo register e che col Demux viene selezionato il registro in cui porre il dato, vediamo come prosegue il circuito. Supponiamo di aver premuto il primo bottone; il dato inviato dal mux finisce nel register. Questo registro invia il dato a tutti i registri (tutti a 2 bit), ma viene salvato in un solo registro, corrispondente all'uscita asserita dal demux in alto. Il funzionamento è il seguente: l'uscita del demux finisce in AND con il segnale di clock (che ha una frequenza abbastanza elevata, ad esempio 64 Hz), ed il risultato finisce nella locazione del clock. In un primo momento avevo collegato direttamente l'uscita del demux nella locazione del clock, ma ho notato che il dato non veniva salvato, e questo perché il dato non riusciva a giungere in tempo nel registro, dunque ho risolto il problema semplicemente ritardando il "segnale di clock" (ovvero l'uscita del demux) ponendolo in AND col clock. Il funzionamento degli altri registri è analogo. A tutti i registri inoltre arriva il segnale di Reset all'entrata "0", in modo da resettare appunto i valori in caso di errore: si noti che il reset è unico e non è possibile resettare un singolo registro.

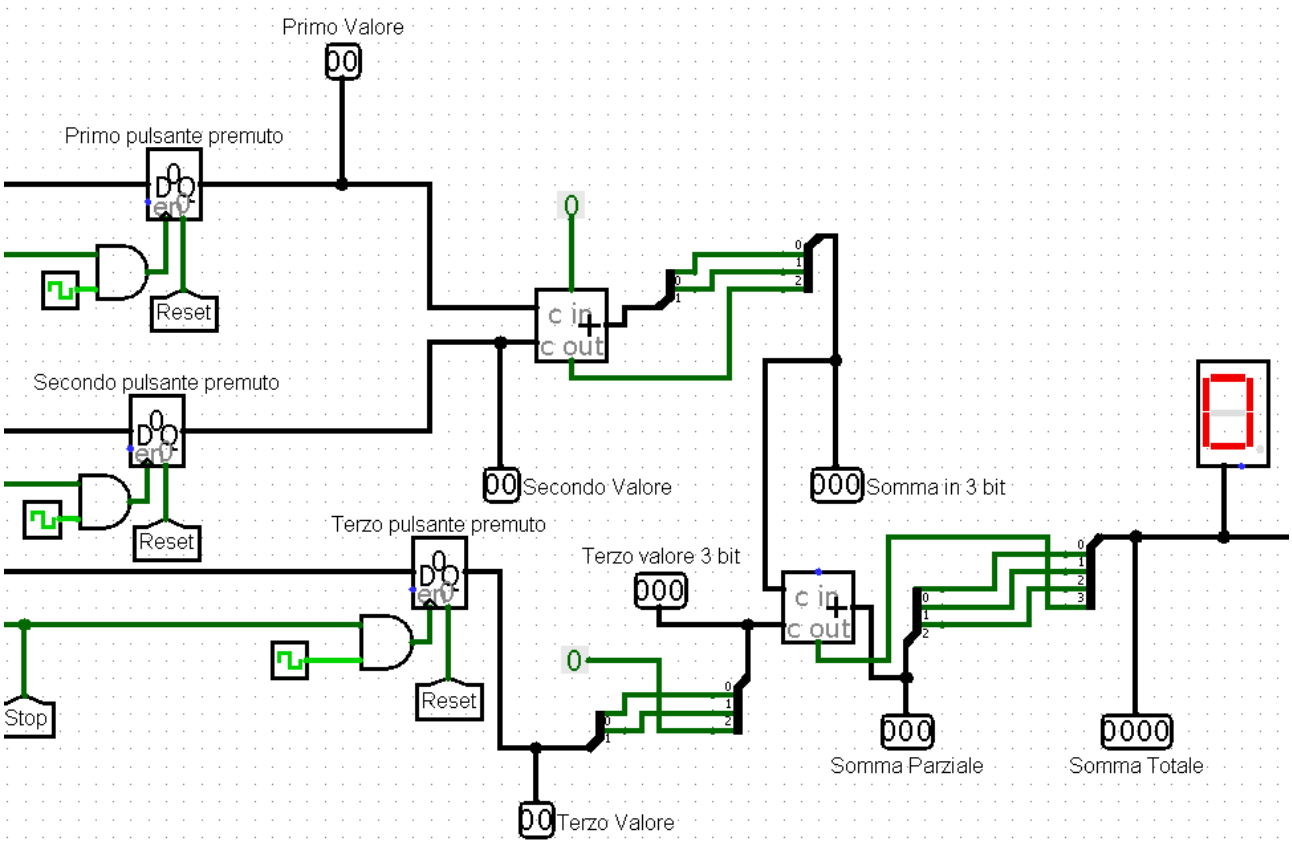
Il valore ora è proiettato in uscita; devo sommarlo per avere una stima sull'importo da pagare. In un primo momento avevo realizzato il circuito in questo modo:



Ovvero, avevo collegato il valore in uscita da un registro con quello dopo in un Full-Adder, il risultato lo avevo posto in un secondo FA ed il risultato lo avevo *desplittato* e *risplittato* aggiungendo il segnale Carry Out in uscita nella locazione “0”, credendo che la posizione 0 fosse dedicata al MSB della somma.

Mi sono accorto tuttavia che la somma non tornava, andavo incontro a Overflow ed inoltre, anche dopo aver risolto il problema dell’overflow, la somma continuava ad essere errata.

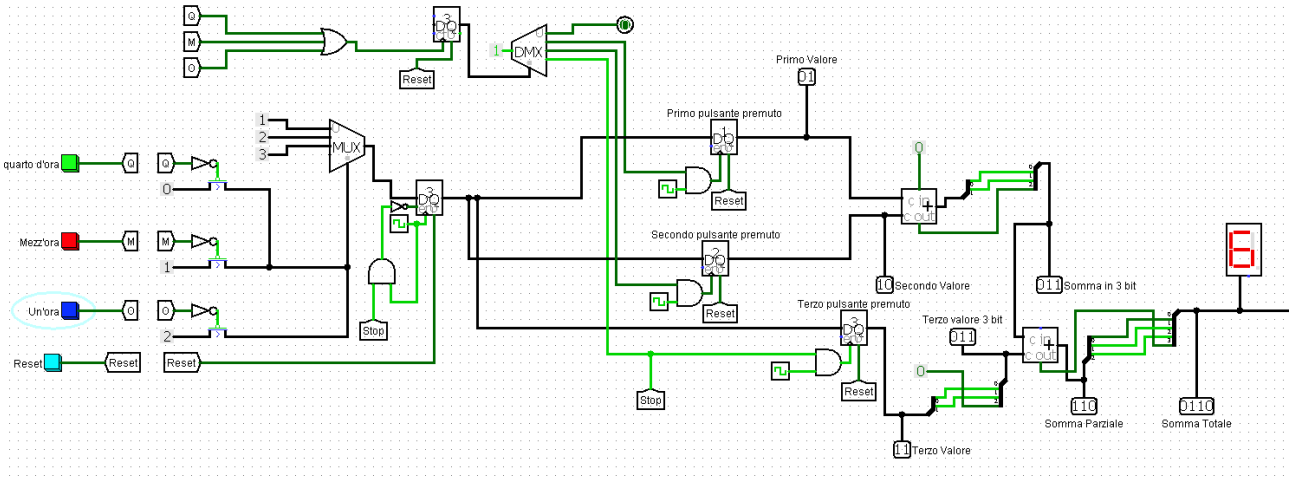
Ho risolto in questo modo il problema:



Ho introdotto dei pin di controllo, sia per comodità mia (mi hanno aiutato a comprendere la natura del problema), sia per comodità del fruitore, per aiutarlo a comprendere cosa stava succedendo nella macchina. Consideriamo la prima somma: una volta sommati i due numeri, il MSB della somma è, chiaramente, il carry out del FA; tuttavia la locazione in cui deve essere posto non è la “0”, ma l’ultima, la “2” in questo caso! Quando inoltre splitto un numero, il suo MSB sarà quello nell’ultima locazione (sempre prendendo ad esempio la prima somma, la posizione “1” dunque). Dopo ogni somma, per risolvere il problema dell’overflow, ho posto uno splitter e un desplitter, portando il numero a 3 bit.

Per quanto riguarda la seconda somma qualcosa varia: si può vedere che dato che è proiettato in uscita dal register viene, prima di essere posto nel carry out, aumentato a 3 bit, dato che i FA funzionano a numeri con lo stesso numero di bit. Il numero ovviamente è “aumentato di bit” utilizzando una costante 0 come MSB, dato che consideriamo solo numeri positivi (tutti i nostri FA infatti lavorano in Unsigned). Il numero ottenuto viene dunque sommato alla “somma parziale” precedente; il risultato viene splittato e desplittato per ottenere il risultato corretto.

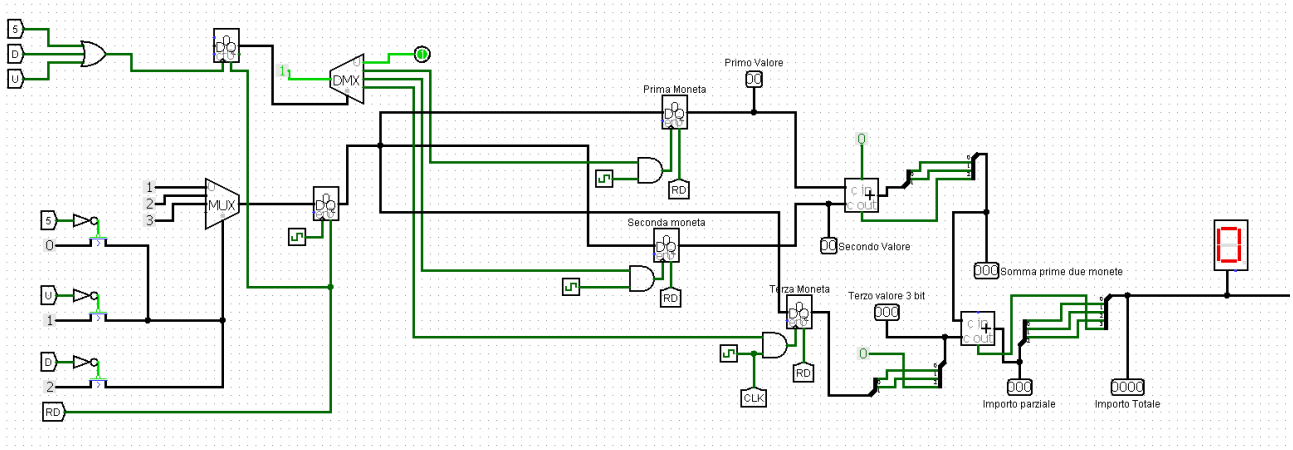
La somma totale è pronta; questa è a 4 bit e finisce nel display a esadecimale, che permette di vedere il risultato, risultato che rappresenta l'importo da pagare perché il parchimetro entri in funzione. Sotto è riportato un esempio di esecuzione tipica:



Il segnale "Stop":

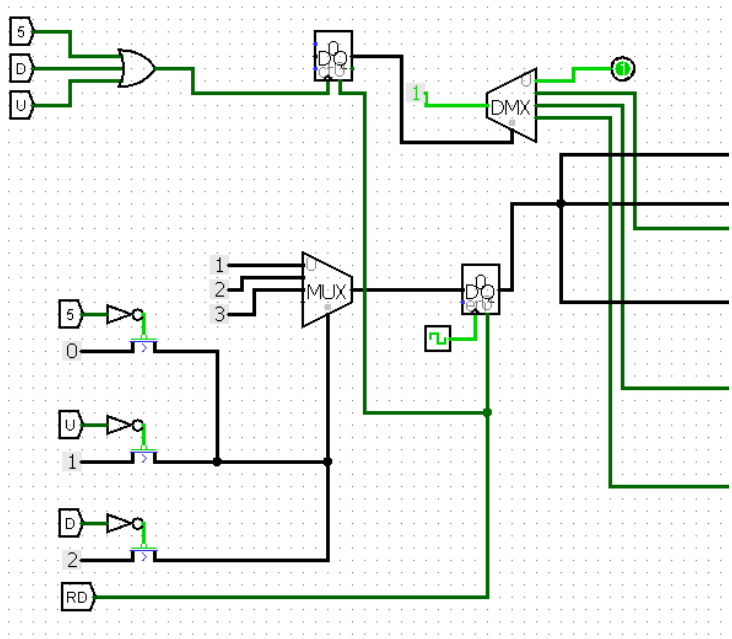
Come si può vedere, il segnale di Stop non è nient'altro che l'ultima uscita del demux. L'ho introdotto per fare in modo che, qualora l'utente cerchi di premere più di 3 volte i bottoni, questo "freezi" il valore nel registro, andando a bloccare il register. Il segnale è messo in AND col clock ed è negato per fare in modo che funzioni, collegare l'uscita del demux direttamente al register non premette il passaggio dell'ultimo valore.

Blocco B

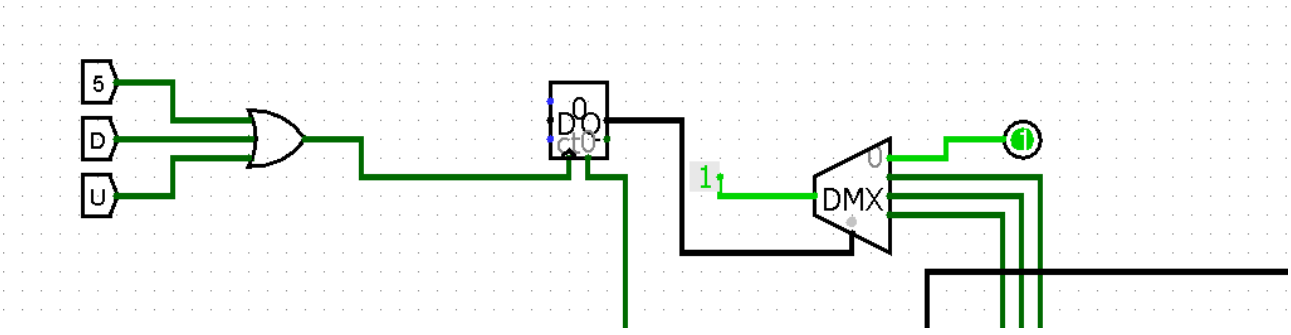


Funzionamento blocco B:

Il blocco B, come già detto, è una sorta di “blocco A” leggermente modificato, di conseguenza anche il suo funzionamento è pressoché identico. Vi sono 3 tunnel, che ricevono il segnale inviato dai rispettivi bottoni (5 – 50 cent, U – un euro, D – due euro). Questo segnale è inviato ad un mux tramite un gioco di transistor identico a quello del blocco A. Dal mux dunque esce il valore corrispondente al bottone premuto, che lo invia a tutti i register.



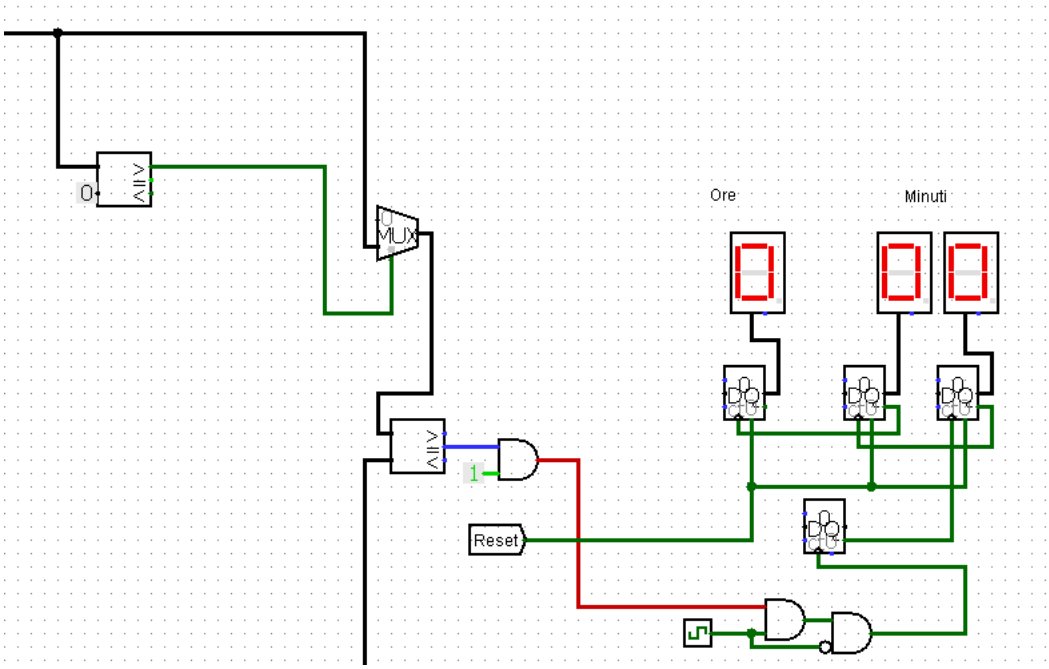
Il valore, ancora una volta, viene salvato utilizzando una porta OR a tre entrate (i vari bottoni, eccetto ovviamente il Restituisci Denaro); l’uscita di questo OR finirà in un Counter ed il numero nel counter finirà come bit di selezione nel Demux. Anche in questo caso la prima porta del demux sarà libera in quanto il counter parte a contare da 0 e dunque è in primis asserita la porta 0.



Il valore del register dunque sarà salvato nei vari register a seconda del bit di selezione del demux.

Il processo di somma è il medesimo di quello del blocco A, dunque si eviterà di trattarlo. Una volta che la somma immessa è pari a quella richiesta, parte in funzione il Blocco C.

Blocco C



Funzionamento blocco C:

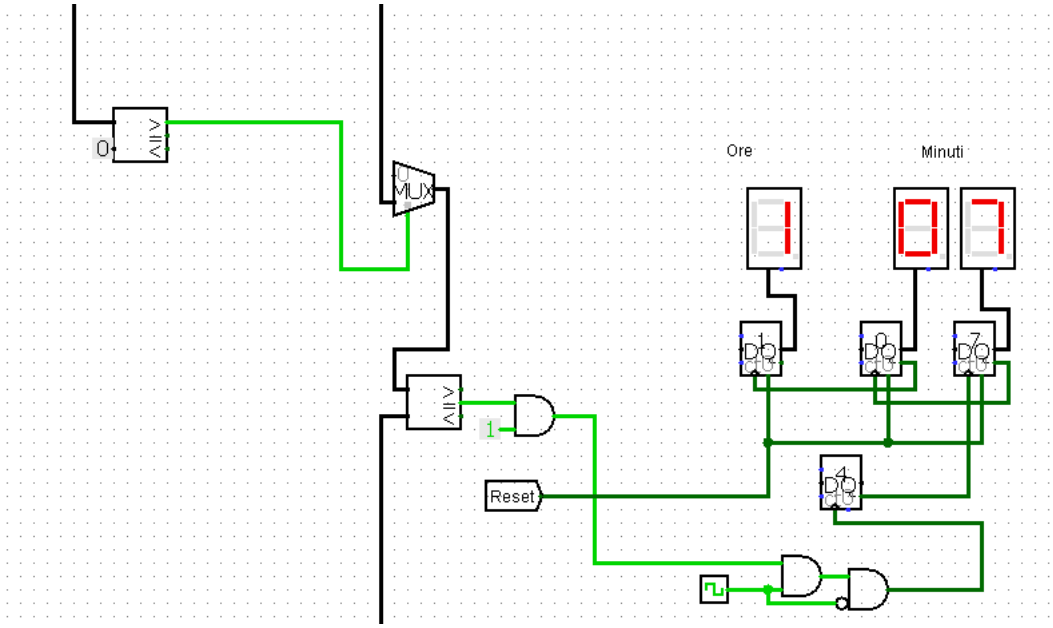
Il blocco C è l'ultima parte circuitale. E' formato da due comparatori, qualche porta AND, 3 counter e 3 display a esadecimali. Il valore di importo da pagare finisce in un comparatore con una costante 0; questo mi serve perché io voglio fare in modo che l'orologio parta quando i due valori sono uguali. E' necessario introdurre un doppio controllo perché altrimenti al momento di avvio i due valori "nei display", essendo entrambi 0 e dunque uguali, andrebbero ad attivare l'orologio. Si risolve appunto il problema ponendo un controllo del primo valore con una costante 0, e dal comparatore in cui sono introdotti i due valori si collega l'uscita di maggioranza (ovvero VALORE-DISPLAY > 0) al bit di selezione di un mux, in cui arriva alla sola entrata "1" il valore nel display. Quando si preme un qualsiasi bottone (eccetto Reset ovviamente), si asserisce l'uscita e viene selezionato il valore. Il

valore esce dal Mux e viene proiettato in un nuovo comparatore, dove arriva anche il valore del secondo display (quello che rappresenta il valore inserito). A questo punto collego l'uscita di UGUAGLIANZA in un AND con una costante "1", per ritardare il valore. Il segnale di questo AND finisce in un nuovo AND con il Clock ed il segnale risultante finisce in un nuovo AND con il Clock negato, per ritardare sempre il valore. Il segnale prodotto finisce nell'input clock di un register che conta fino a 64, prima di asserire poi la sua uscita (per uscita si intende il Carry, che si asserisce quando il counter raggiunge il massimo).

A questo punto vi sono 3 display, tutti lavorano a 4 bit: il primo ha come valore massimo 9, il secondo ha come valore massimo 5 e ha come valore massimo 3. Ogni uscita Carry è collegata all'input clock del successivo. Quando il counter che conta fino a 64 raggiunge il massimo, parte un segnale che fa entrare in azione il counter successivo.

Il primo counter rappresenta i minuti da 0-9, il secondo i minuti "decimali" e quindi va da 0-5, ed il terzo le ore (e conta fino a 3 perché non posso chiedere più di 3 ore di sosta al parchimetro). Il primo counter lavora in "Rising edge", il secondo in "Falling Edge" per evitare che quando il primo raggiunga 9 il secondo si aggiorni immediatamente, ma attenda il fronte di discesa per aggiornarsi. Il terzo, per evitare lo stesso problema, lavora in "Rising Edge".

Esempio di esecuzione:



Considerazioni finali ed problemi:

Il circuito permette dunque di simulare il funzionamento di un parchimetro, per quanto molto semplificato.

Vi è un problema che non sono riuscito a risolvere; il parchimetro non si ferma all’orario richiesto. Ho pensato ad una soluzione che tuttavia non sono riuscito a mettere in pratica: ci deve essere un segnale che capta a seconda del bottone quante volte viene premuto (dunque anche un counter è necessario). A questo punto ad ogni counter deve essere associato un valore (per i minuti 9, per i “minuti decimali” 59, per le ore 1) e collegare questo valore all’orologio invece dell’importo.