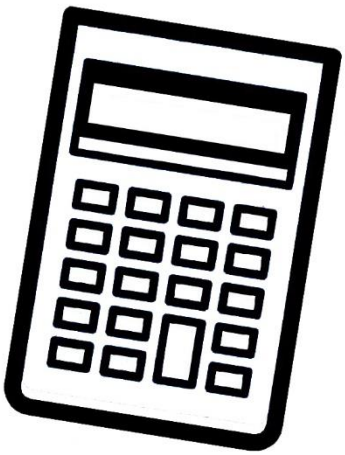
UNICALC

PROGETTO ARCHITETTURA DEGLI ELABORATORI 2021/2022

INFORMATICA APPLICATA E DATA ANALYTICS (IADA) Cagliari
A cura di Davide Senette Matricola N. 60/79/00052

Indice progetto calcolatrice



1. IDEA E FUNZIONAMENTO

L'idea del progetto è quella di costruire una calcolatrice digitale, attraverso l'uso delle porte logiche (AND, OR, NOT, XOR, etc.) e con l'ausilio delle tabelle di verità per ricavare circuiti logici in grado di risolvere in notazione binaria operazioni di somma, sottrazione e moltiplicazione.

I dati verranno inseriti attraverso una tastiera a 5 bit in binario e visualizzati a schermo tramite 2 BCD, che con un opportuno chip convertirà gli input del contatore da binario a decimale. Essendoci solo due BCD l'output massimo visualizzabile è 99.

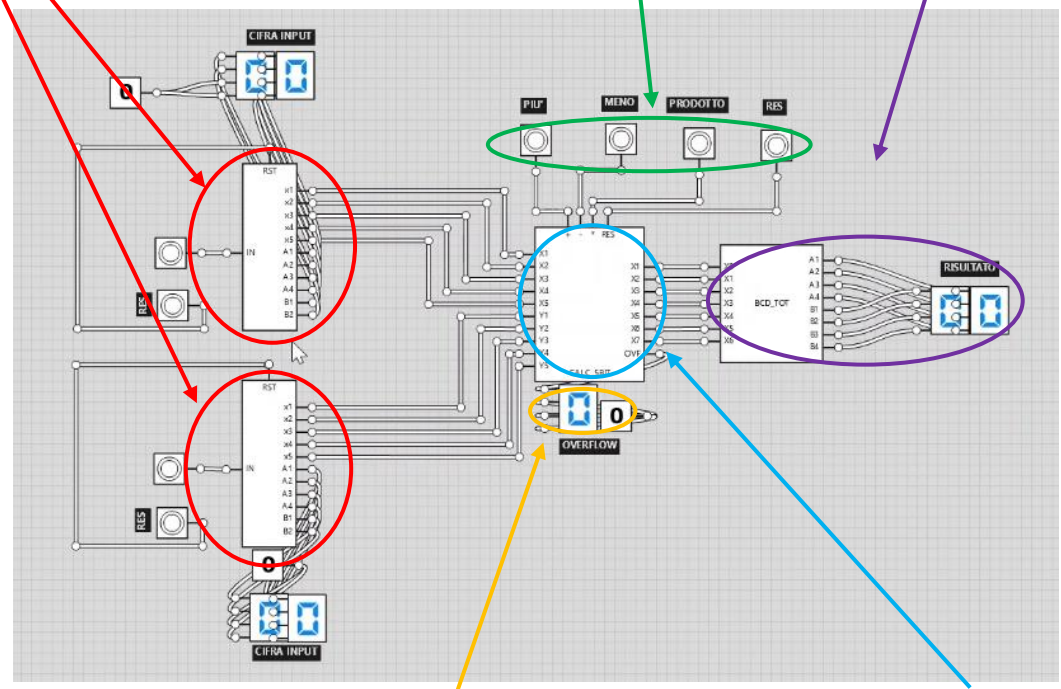
I dati in formato binario provenienti dalla tastiera vengono inseriti come input nel chip di calcolo che in base all'operazione da eseguire, decisa dall'utente, effettuerà le dovute operazioni restituendo in output il risultato. Tutte le operazioni sono eseguite in 5 bit. La moltiplicazione lavora in 5 bit, ma per scelta puramente visiva restituisce un output a 7 bit. Il chip di calcolo possiede anche un piedino di controllo per l'overflow, che va a collegarsi ad un altro BCD, impostandolo a 1, per informare l'utente che il risultato non è valido a causa di uno sforamento della memoria di calcolo allocata.

Infine, l'output del chip di calcolo viene collegato ad un altro circuito integrato identico a quello all'interno della tastiera, il cui scopo è quello di convertire i segnali in formato binario, dati in input, in determinati output in grado di controllare come un unico blocco i due BCD per restituire il numero in decimale a schermo. L'output sarà visibile anche dopo il calcolo finché l'utente non decide di cancellare il risultato ed eseguire un'altra operazione, cioè la calcolatrice sarà provista anche di una memoria interna in grado di conservazione le informazioni di input e di output inserite dall'utente.

Tastiera Input

SELEZIONE OPERAZIONI

CHIP OUTPUT



Chip di calcolo

Bit Overflow

2. CHIP DI CALCOLO

Come accennato nella sezione introduttiva, la nostra calcolatrice dovrà svolgere tre delle principali operazioni aritmetiche, ovvero la somma la sottrazione e la moltiplicazione. Queste operazioni saranno effettuate solo con numeri interi e solo a 5 bit, cioè con un numero massimo di 31 per input (2^5 -1 = 31), e il risultato sarà anch'esso di 5 bit, per la somma e la sottrazione, mentre la moltiplicazione per motivi puramente estetici potrà restituire in uscita 7 bit anziché 5, il calcolo avverrà in ogni caso con due numeri in input a 5 bit. Il risultato verrà memorizzato attraverso 8 flip-flop D (5 per somma e sottrazione, +2 per il prodotto, +1 per il bit di over-flow).

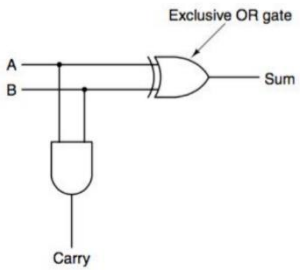
Ora vediamo nel dettaglio la costruzione dei circuiti logici, la rispettiva tabella di verità ed infine l'implementazione degli input di controllo per la gestione della scelta dell'operazione aritmetica o del reset del risultato.

- **SOMMA**
  Per costruire il circuito logico di somma ho utilizzato un half-adder come circuito per i primi due bit, perché non è necessario considerare alcun riporto iniziale.

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tabella di verità half-adder.

Il circuito logico half-adder è costituito da una porta XOR e una porta AND collegati ad A e B. XOR effettua la somma vera e propria e restituisce 1 solo quando i due bit in ingresso sono diversi, altrimenti restituisce 0. AND verifica la presenza di un riporto e restituisce 1 sono quando sia A sia B valgono 1, in tutti gli altri casi restituisce 0.

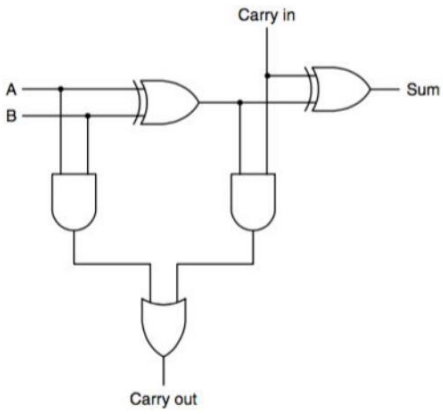


Per effettuare le somme successive dei restanti 4 bit non possiamo usare il circuito half-adder, perché non considera i riporti di eventuali operazioni di somma precedenti. Per considerare i riporti precedenti ho usato il full-adder.

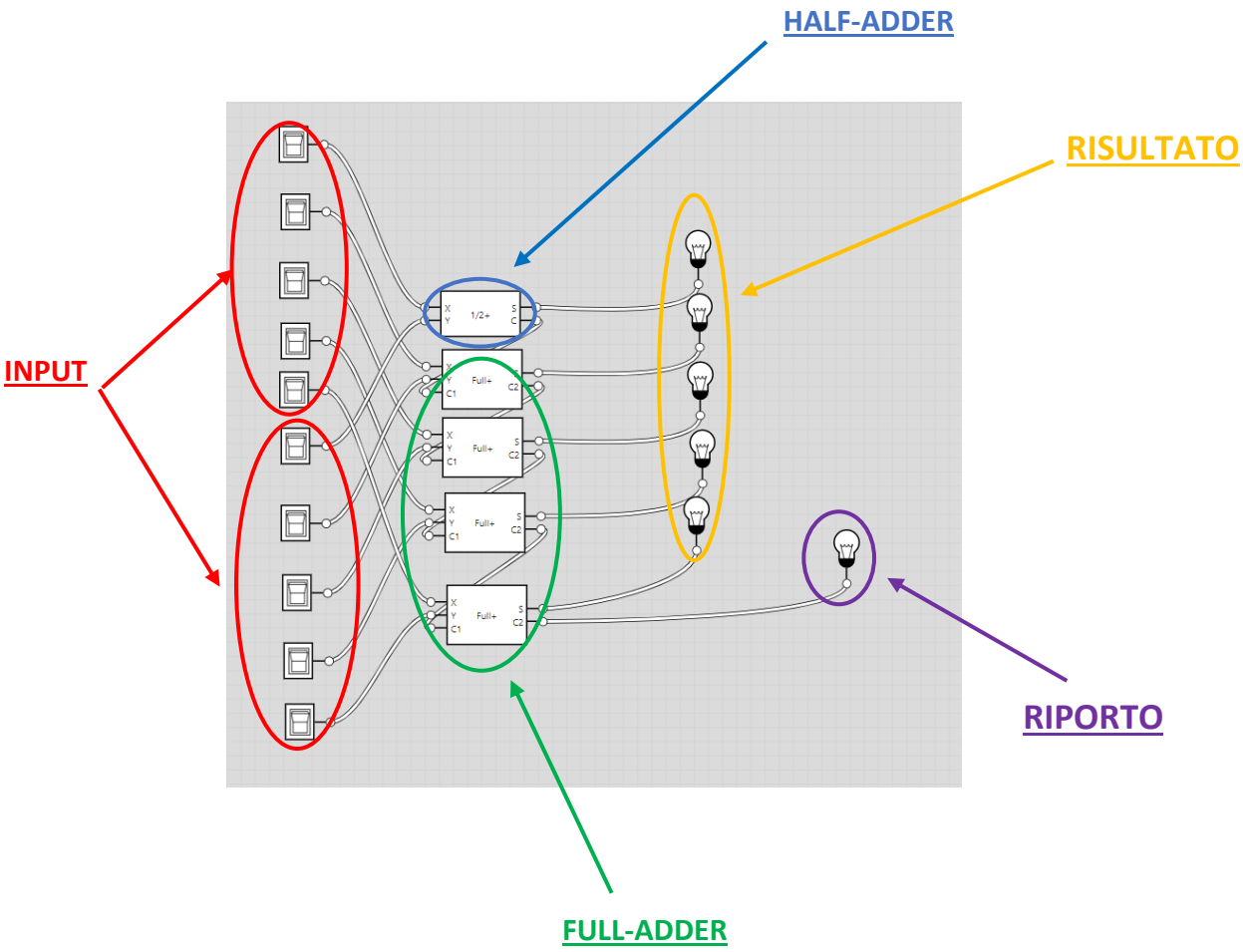
A	B	Carry in	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabella di verità del full-adder.

La prima porta XOR calcola la somma di A e B, alla quale viene collegata anche una porta AND per verificare la presenza di un eventuale riporto. L'output della prima XOR viene collegato in input ad un'altra porta XOR che calcola la somma con il riporto in input e verificando anche qui che sia presente o meno un riporto con un'altra porta AND. Gli output delle due porte AND vengono infine collegate ad una porta OR che restituisce il riporto della somma totale di A B e il riporto iniziale di input.



Dopo aver costruito i due circuiti si vanno a collegare i vari bit di input del nostro chip di somma a 5 bit. In particolare, i primi due si collegano al chip HALF-ADDER, i restanti 4 bit sono collegati al chip del FULL-ADDER che riceverà a cascata il riporto dell'operazione precedente. L'ultimo riporto, ovvero del quarto FULL-ADDER è il riporto della nostra somma che verrà collegato in uscita come bit di over-flow al BCD.



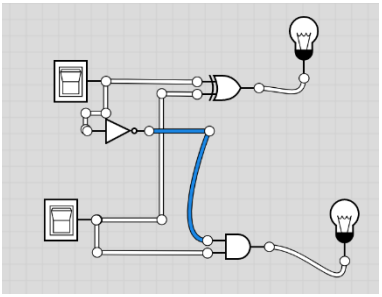
•SOTTRAZIONE

La sottrazione funziona in modo molto simile alla somma. Abbiamo bisogno di un circuito che effettua la sottrazione senza riporto nell'operazione, e uno che va a considerare anche il riporto.

A	B	SOTTRAZIONE	RIPORTO
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Tabella di verità della semi-sottrazione.

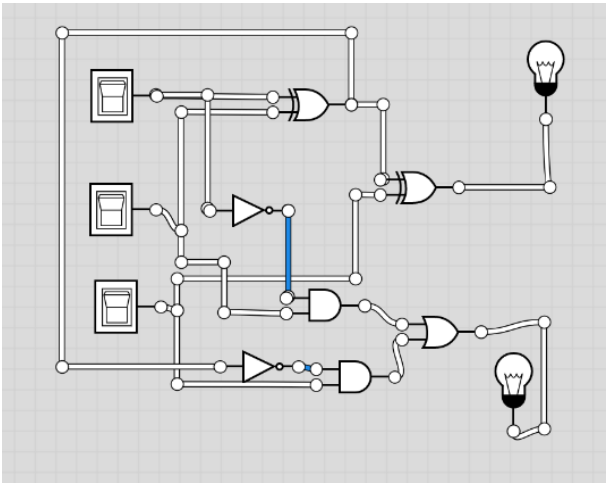
Circuito semi-sottrazione corrispondente.

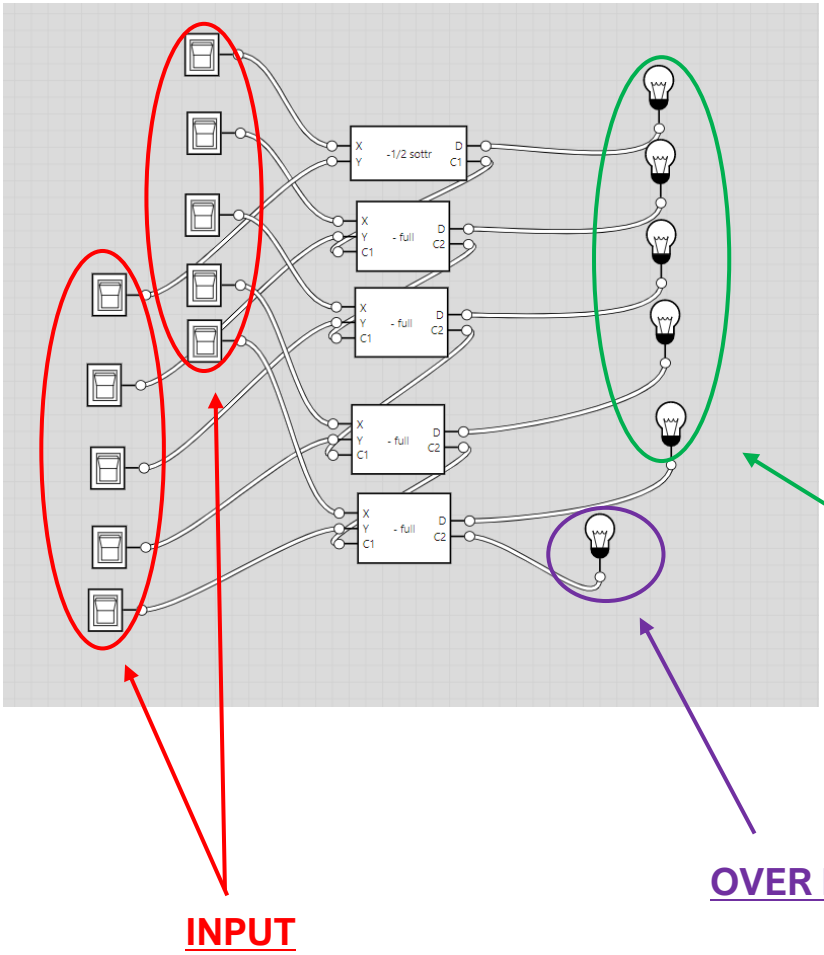


A	B	C1	S	C2
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Tabella di verità della sottrazione con riporto. A e B sono i numeri da sottrarre, C1 è il riporto in ingresso, S è il risultato della sottrazione e C2 è il riporto in uscita.

Circuito sottrazione completa con riporto.





Come nell'addizione anche nella sottrazione i primi due bit saranno gestiti dal circuito di semi-sottrazione, mentre i bit successivi dal circuito di sottrazione completo.

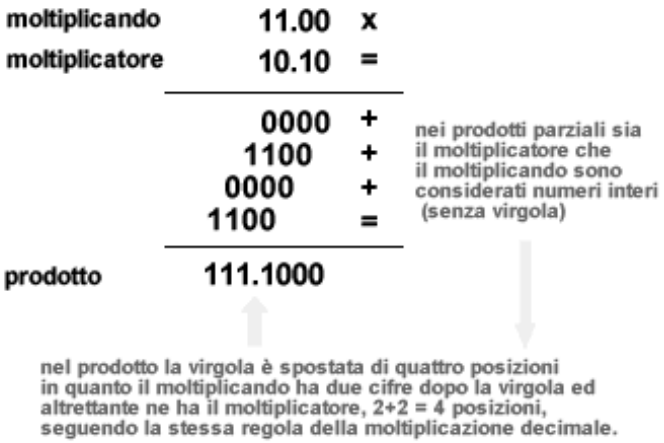
- **MOLTIPLICAZIONE**

L'ultima operazione aritmetica della calcolatrice è la moltiplicazione. Scriviamo la tabella di verità e facciamo alcune considerazioni.

A	B	A*B
0	0	0
0	1	0
1	0	0
1	1	1

Come si può notare la tabella di verità della moltiplicazione tra due bit corrisponde alla tabella di verità della porta logica AND, ed effettivamente per fare il prodotto tra due bit basta una semplice AND.

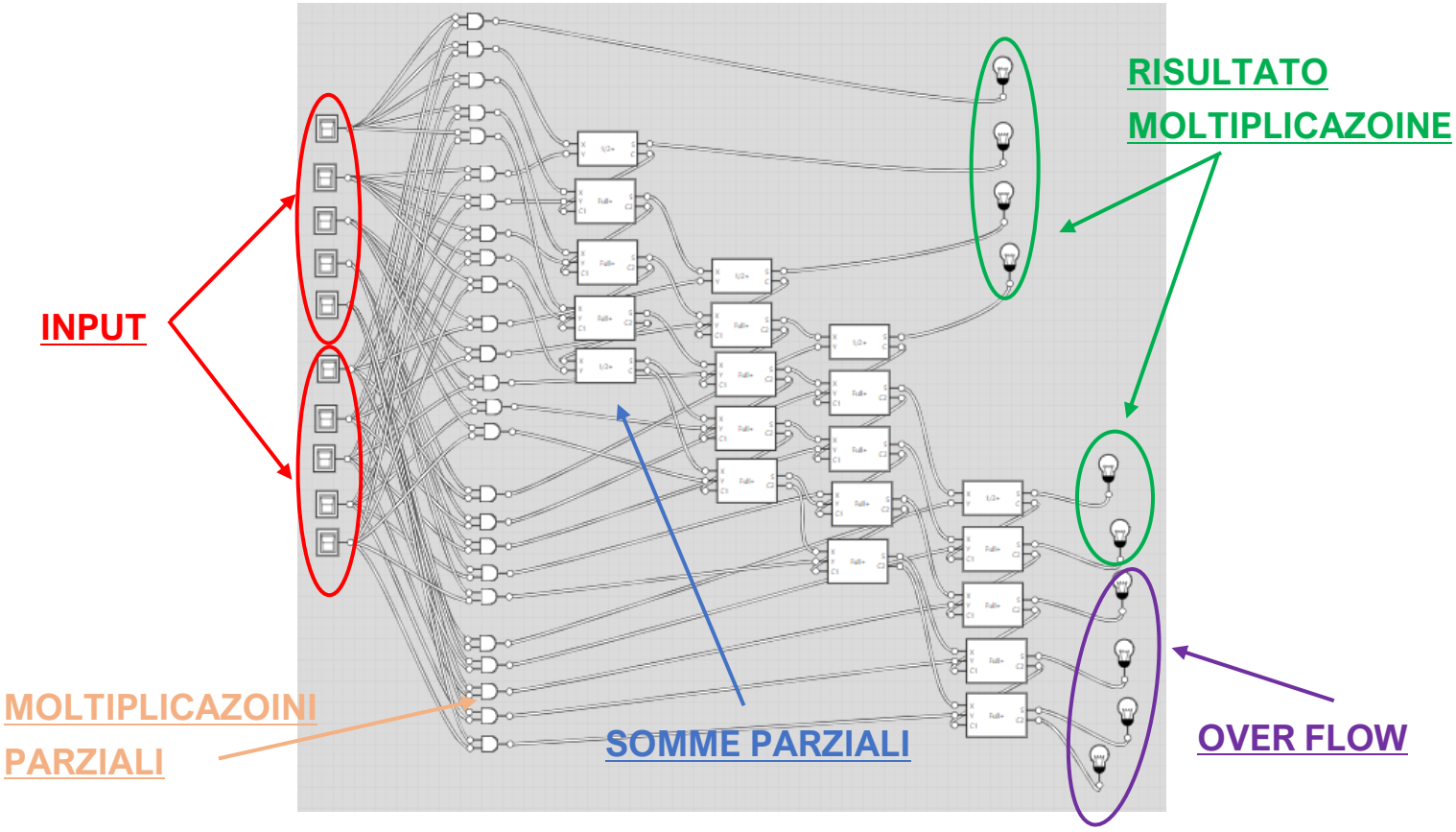
Quando abbiamo più input la situazione si complica, infatti oltre al calcolo della "moltiplicazione parziale", tra i vari bit, il risultato dopo ogni riga va spostato di una posizione, ed infine va sommato il tutto.



Per semplificare l'operazione la soluzione che ho adottato per sommare più righe è stata quella di eseguire una serie di somme parziali.

Ovvero le prime due righe vengono sommate, e il risultato verrà successivamente sommato con la terza riga e così via.

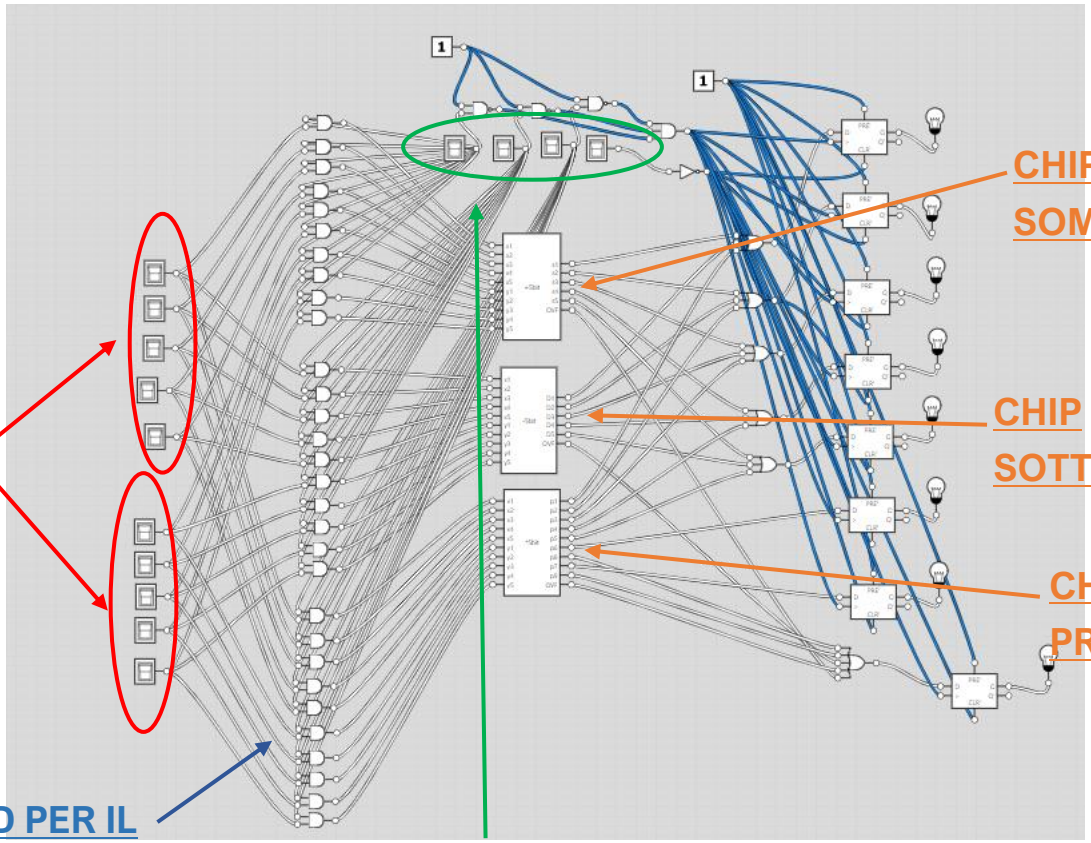
Il circuito risultante sarà composto perciò da una serie di porte logiche AND che eseguono una moltiplicazione parziale di ogni bit del moltiplicatore con ogni bit del moltiplicando. Infine, si sommano i risultati come spiegato prima, utilizzano i circuiti half-adder e full-adder.



- **GESTIONE SCELTA**

Per gestire l'operazione da eseguire dobbiamo costruire un nuovo circuito logico che comprendo le nostre operazioni, ovviamente, ma con un sistema che gestista l'operazione da calcolare. Se voglio eseguire la somma i circuiti di sottrazione e moltiplicazione non devono ricevere alcun input in ingresso, in modo tale da non falsare /corrompere il risultato finale della nostra operazione.

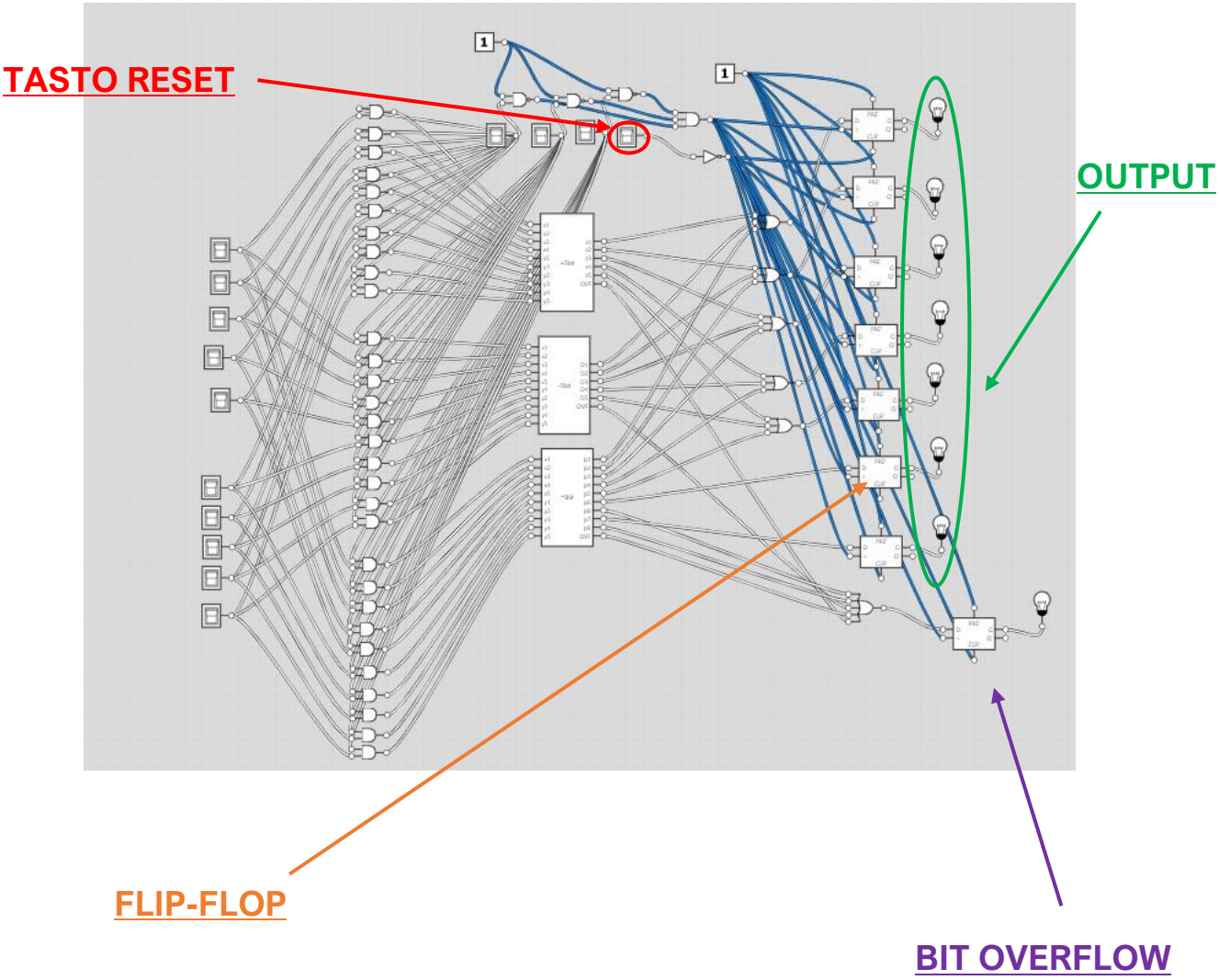
Per risolvere questo problema semplicemente possiamo usare una serie di porte logiche AND, in modo tale da disattivare gli ingressi ai chip di calcolo da non utilizzare, in base alla scelta in input decisa dall'utente.



- **MEMORIA**

Per memorizzare i valori risultanti dalle nostre operazioni aritmetiche e non perderli dopo aver cliccato il bottone, quello che possiamo fare è inserire alcuni flip-flop di tipo D.

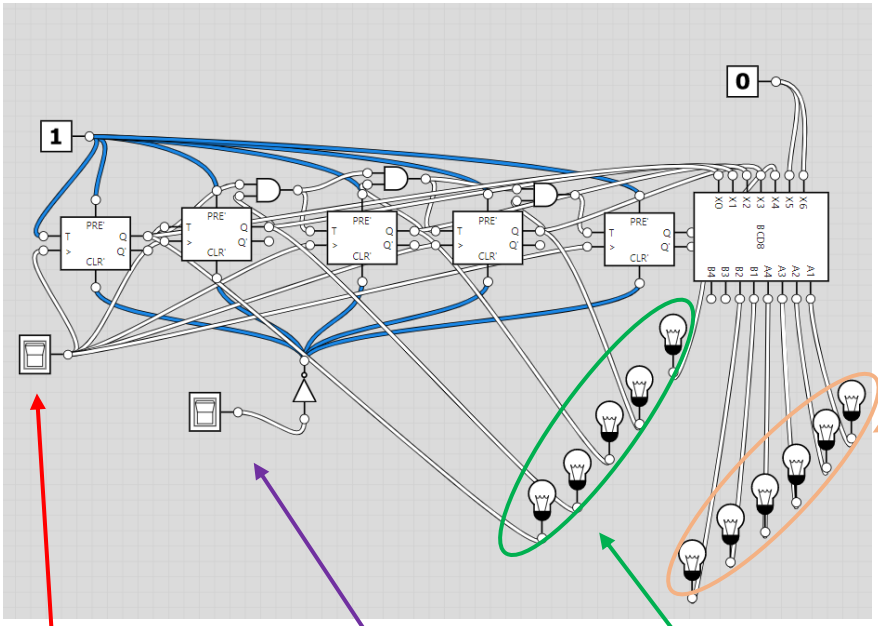
I primi 5 flip-flop sono collegati ai primi 5 bit delle porte logiche OR e fanno riferimenti ai risultati di somma moltiplicazione e sottrazione, i restanti 2 flip-flop sono collegati solo al chip di moltiplicazione e l'ultimo per il bit di overflow. Quando le linee del risultato si attivano, quindi hanno stato 1, funzionerà come segnale di clock attivando il flip flop e facendo cambiare stato all'output Q che memorizza il risultato, da 0 a 1, preservandolo finché non si preme il tasto reset, che è il nostro quarto tasto di scelta per l'utente finale, che azzera tutti gli 8 flip-flop reimpostando Q a 0.



- **Tastiera**

Per creare la tastiera ho utilizzato un Ripple-Counter, che è un tipo speciale di contatore asincrono in cui l'impulso di clock si propaga attraverso il circuito.

Il contatore è costituito da 5 flip-flop di tipo T, per rappresentare tutti i numeri in binario da 0 a 31. L'impulso di clock esterno viene applicata solo al primo flip-flop, mentre la sua uscita funzionerà da clock per il successivo e così via. Ad ogni clock si fa variare lo stato dei flip flop di un bit, in ordine crescente. Quindi inizialmente avremo 00000, dopo un clock il 1° flip-flop cambierà stato a 1, i 5 bit diventano quindi 00001 = 1, con un altro clock il 1° flip-flop si azzera, ma funziona da clock per il 2° attivandolo e facendo cambiare il suo stato da 0 a 1, i 5 bit diventano 00010 = 2. Quando si raggiunge il valore MAX di 31, ovvero 11111, il contatore si resetta e riparte dallo stato iniziale, ovvero tutti i flip-flop saranno 00000 = 0.



SEGNALI DI CONTROLLO DEL BCD PER OUTPUT DECIMALE

CLOCK

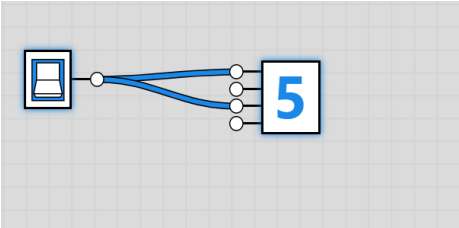
TASTO RESET

OUTPUT IN BINARIO

Il numero presente nel contatore viene convertito in base dieci dal chip BCD, in alto sulla destra. Il circuito integrato finale avrà sia il numero in binario da passare alla nostra ALU, sia i bit da passare al BCD per stampare a schermo il numero digitato.

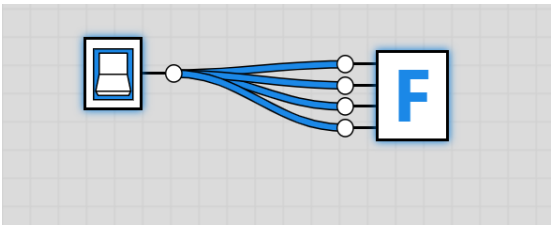
- ## Chip per la gestione dell’output

Come visto precedentemente, la nostra calcolatrice lavoro con somme sottrazioni e moltiplicazioni in forma binaria, e anche i risultati delle operazioni sono in forma binaria. Per convertire inizialmente i nostri dati da binario a qualcosa di più visivo e comprensibile possiamo usare un chip per la gestione dei bcd.



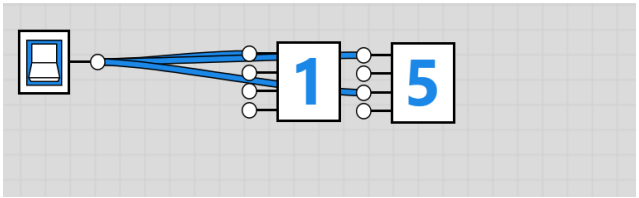
Il funzionamento del bcd è molto semplice, è composto da 4 piedini che sono gli input. Ogni piedino rappresenta una potenza di 2, quindi il primo partendo dall’alto equivale a 2^0 poi 2^1,2^2 e infine 2^3. Variamo l’accensione dei piedini, otteniamo i numeri da 0 a 15, cioè in formato esadecimale.

Se vogliamo gestire due BCD e solo con output in base 10 le cose si complicano. Se abbiamo 15 non vogliamo come output F, bensì 1 nel bcd sulla sinistra e 5 nel bcd sulla destra.



Come il 15 viene rappresentato normalmente in un bcd. F in esadecimale è 15.

Come vorremo gli output dei nostri valori.



Quindi il primo passo è quello di costruire una tabella di verità che contenga tutti i segnali di ingresso del nostro numero in binario da una parte, e dall'altra gli input predisposti per avere in output i nostri risultati stampati in base 10. La tabella avrà una dimensione in altezza pari a tutti i numeri che vogliamo rappresentare ovvero 100(va dal numero 0 al numero 99).

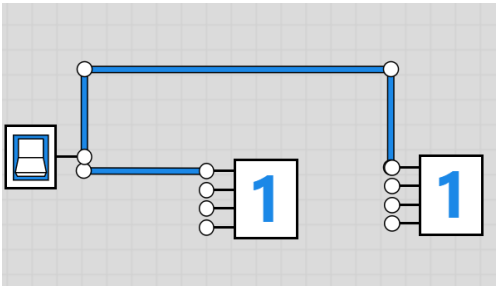
X7	X6	X5	X4	X3	X2	X1	B4	B3	B2	B1	A4	A3	A2	A1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	1	1	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	1	0	0	0	0	0	1	0	1
0	0	0	0	1	1	0	0	0	0	0	0	1	1	0
0	0	0	0	1	1	1	0	0	0	0	0	1	1	1
0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	0	0	1	0	0	1	0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	1	0	0	0	1	0	0	0	1
0	0	0	1	1	0	0	0	0	0	1	0	0	1	0
0	0	0	1	1	0	1	0	0	0	1	0	0	1	1
0	0	0	1	1	1	0	0	0	0	1	0	1	0	0
0	0	0	1	1	1	1	0	0	0	1	0	1	0	1
0	0	1	0	0	0	0	0	0	0	1	0	1	1	0
0	0	1	0	0	0	1	0	0	0	1	0	1	1	1
0	0	1	0	0	1	0	0	0	0	1	1	0	0	0
0	0	1	0	0	1	1	0	0	0	1	1	0	0	1
0	0	1	0	1	0	0	0	0	1	0	0	0	0	0
0	0	1	0	1	0	1	0	0	1	0	0	0	0	1
0	0	1	0	1	1	0	0	0	1	0	0	0	1	0
0	0	1	0	1	1	1	0	0	1	0	0	0	1	1
0	0	1	1	0	0	0	0	0	1	0	0	1	0	0
0	0	1	1	0	0	1	0	0	1	0	0	1	0	1
0	0	1	1	0	1	0	0	0	1	0	0	1	1	0
0	0	1	1	0	1	1	0	0	1	0	0	1	1	1
0	0	1	1	1	0	0	0	0	1	0	1	0	0	0
0	0	1	1	1	0	1	0	0	1	0	1	0	0	1

le celle da X1 a X7 rappresentano il nostro numero binario.

Mentre le celle da B1 a B4 i piedini di controllo del bcd sulla sinistra e da A1 a A4 i piedini di controllo del nostro bcd sulla destra.

Prendiamo in esame questa riga, i valori da X1 a X7 sono 0001011 ovvero 8+2+1=11 in forma binaria. Quello che vogliamo ottenere è quindi l'accensione solo di B1 e A1 che rappresentano nei due bcd 1 e 1.

Ovvero in questo modo:



Quindi nella tabella di verità inseriremo 1 in B1 e A1 ovvero i bit del bcd che dobbiamo attivare, in questo caso, per formare il numero 11.

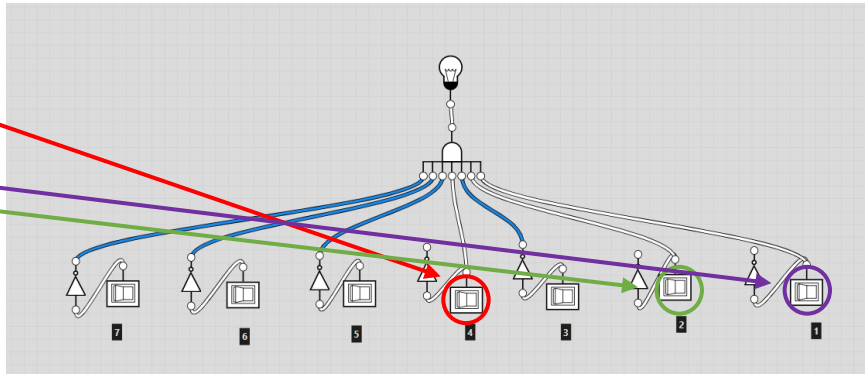
Per la costruzione del circuito si seguono le seguenti regole:

- Ci si munisce di invertitori per generare la negazione di ciascun input
- Si utilizza una porta AND per ciascun termine il cui valore nella colonna risultato sia 1
- Si collegano le porte AND agli input appropriati

Se prendiamo in esame l'esempio di sopra per il numero 11, come si vede nella tabella dobbiamo attivare il pin A1 quindi costruiamo il blocco di controllo:

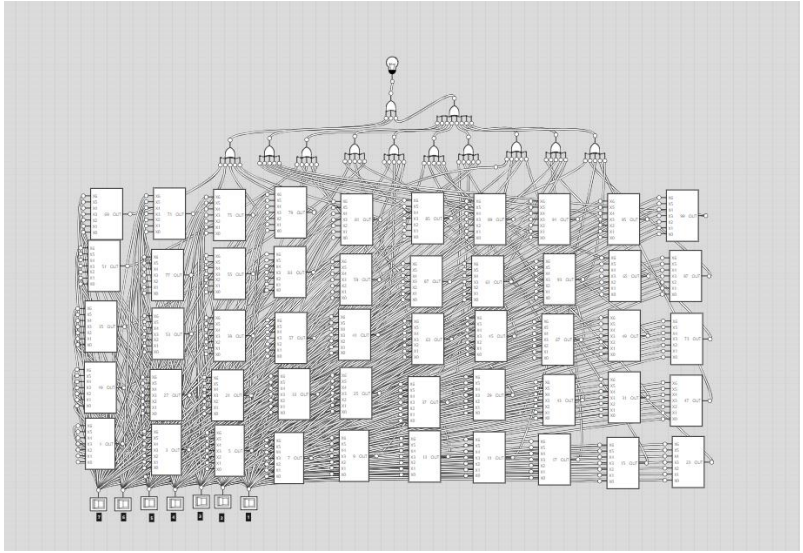
11 = 0001011 <---cifra in binario

Quindi si nega tutto tranne X4, X2 e X1.

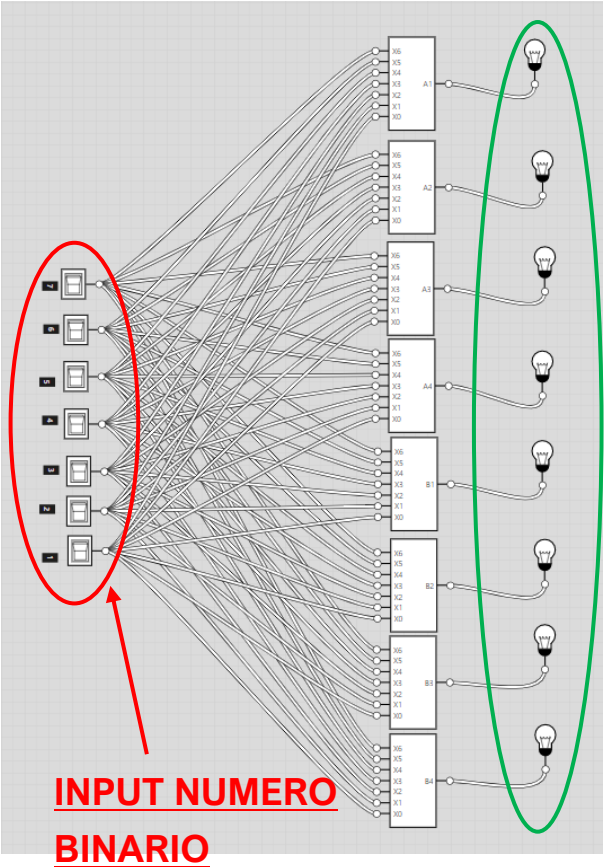


Si collega tutto alla porta AND e si restituisce il risultato della porta come output, rappresentato dalla lampadina, che verrà attivata quando l'utente digiterà 11 nella tastiera.

Si procede a blocchi per ogni per ogni risultato a 1 nella colonna, di ogni pin di controllo del bcd A e B.



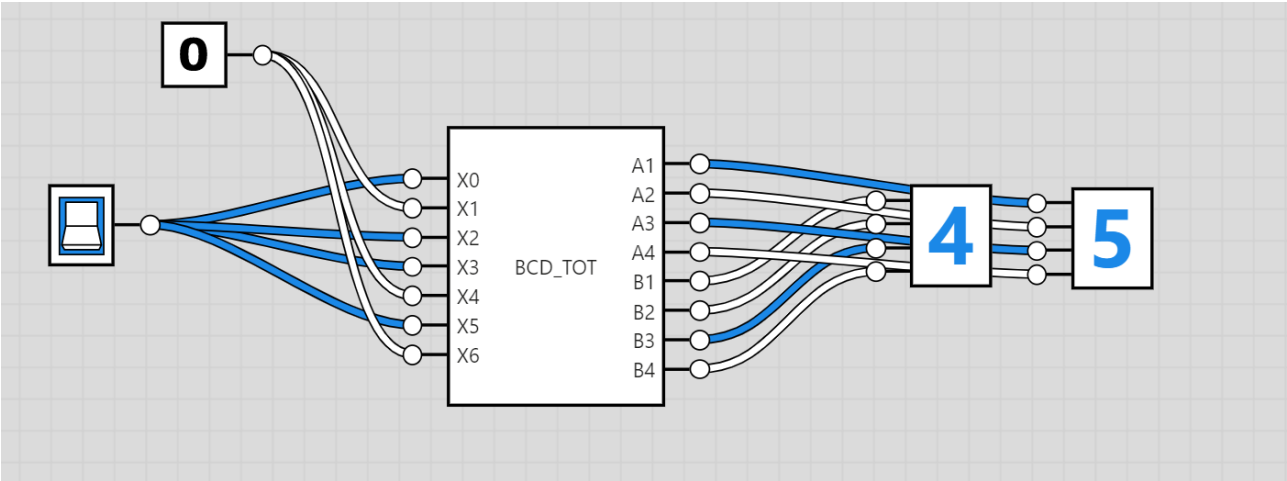
Nell'immagine di fianco, ho riportato il circuito completo per il controllo del solo bit di A1. Ogni risultato dei blocchi viene collegato ad una porta OR e mandato il segnale in output.



Il risultato finale è l'insieme dei chip di controllo per ogni sezione dei bcd A e B, che ricevono in input i valori in binario e attivano i rispettivi bit per visualizzare il risultato in decimale.

OUTPUT PER IL CONTROLLO DEI SINGOLI PIEDINI DEL BCD

In questo esempio, ho inserito come cifra di prova 0101101 come input del nostro circuito totale, che corrisponde in decimale a 45, e collegato i rispettivi bit di output ai rispettivi ingressi in input del bcd.



3. POSSIBILI MIGLIORAMENTI E LIMITI

Alcuni possibili miglioramenti che possono essere implementati, e limiti presenti che andrebbero risolti nella calcolatrice sono:

Aggiungere una memoria più grande per permettere all'utente la possibilità di svolgere operazioni con numeri molto più grandi di quelli concessi attualmente. La somma non permette con 5 bit di ottenere numeri più grandi di 31, la sottrazione invece non permette di ottenere output negativi, con una memoria maggiore potrebbe essere implementata una codifica con il complemento a 2 dei numeri, senza limitarsi all'insieme dei numeri naturali.

L'implementazione del risultato della moltiplicazione a 7 bit, fatta per motivi puramente estetici, avendo l'output massimo a 99 con i bcd, causa alcuni problemi per valori che hanno come prodotto un valore compreso tra 100 e 127 non restituendo l'overflow come segnale e quindi stampando valori scorretti. Si può risolvere questo problema implementando una memoria maggiore e aggiungendo un terzo bcd per avere risultati in output da 0 a 999.

Inoltre, è assente l'operazione di divisione, principalmente per difficoltà implementative che ho riscontrato personalmente. Infatti, trovare i valori con cui applicarla richiederebbe eseguire un ciclo generalizzato per capire quante volte il divisore "ci stà" nel dividendo, poi sottrarlo e ripete l'operazione finché il divisore non risulta essere più grande del dividendo, ottenendo alla fine il resto e il quoziente. Un modo più semplice che si potrebbe implementare per eseguire **alcune** divisioni è quella dello shift a destra, che consente di effettuare divisioni solo per due, questa è una grossa limitazione perché ad esempio non posso dividere per 3 o per 5.

Esempio: (18) B.10 = (1010) B2 → SHIFT A DESTRA → (0101) B.2 = (5) B.10

Un ulteriore limite della divisione tramite shift è il problema della perdita del resto, per tutti i numeri dispari. Infatti, l'operazione di shift non prevede la conservazione del valore di resto, essendo solo una traslazione dei bit. Se non si volesse considerare il resto andrebbe comunque bene per dividere numeri interi per 2.

Esempio (5) B.10 = (0101) B.2 → SHIFT A DESTRA → (0010) B.2 = (2) B10 il resto 0.5 si perde nell'operazione di shift.

Link al video della calcolatrice all'opera:

https://youtu.be/zGFz0-3X-FE