Vediamo un ultimo esempio di circuito numerico: il circuito moltiplicatore.

U = a\*b

Se a, b sono cifre binarie il loro prodotto corrisponde a una funzione AND:

|  |  |  |
| --- | --- | --- |
| a | b | a\*b |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

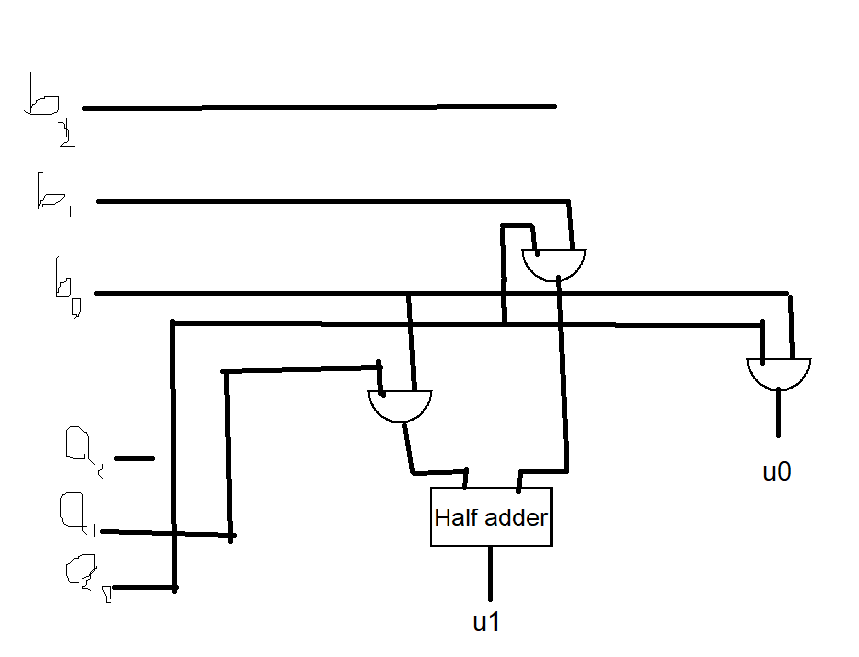
Se a, b sono rappresentati su 2 bit:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a1a0\b1b0 | 00 | 01 | 11 | 10 |
| 00 | 0000 | 0000 | 0000 | 0000 |
| 01 | 0000 | 0001 | 0011 | 0010 |
| 11 | 0000 | 0011 | 1001 | 0110 |
| 10 | 0000 | 0010 | 0110 | 0100 |

U3 = a1\*a0\*b1\*b0

U2 = b1\*a1\*(-b0+-a0)

U1 = …



Come si vede già a 2 bit è abbastanza complessa. Si può quindi cercare un metodo per ottenere una realizzazione modulare. Per farlo ricorriamo alla moltiplicazione in colonna (le buone vecchie tabelline porcodio).

111\*

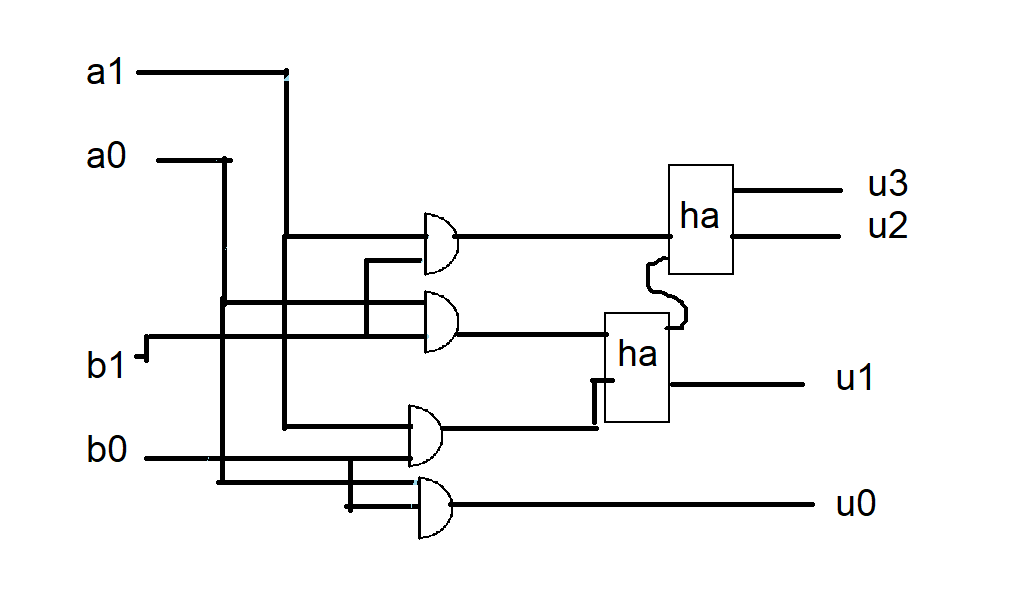
101=

111+

0000+

11100=

100011



A 4 bit

A3 a2 a1 a0 \*

B3 b2 b1 b0 =

A3b0 a2b0 a1b0 a0b0 +

A3b1 a2b1 a1b1 a0b1 - +

A3b2 a2b2 a1b2 a0b2 - - +

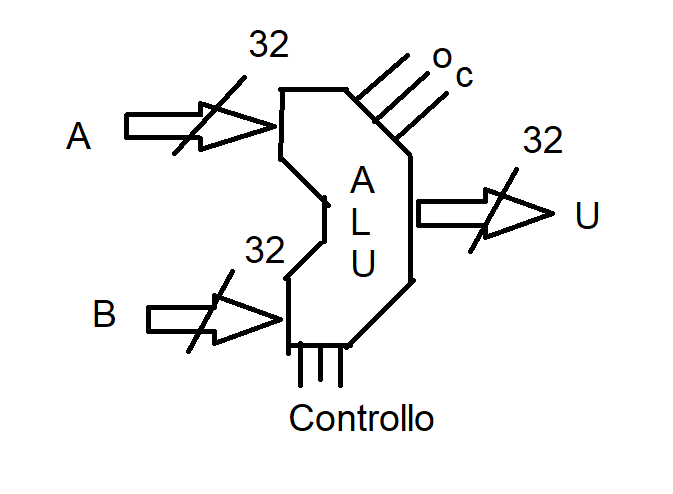
A3b3 a2b3 a1b3 a0b3 - - - =

Calcolare i riporti in questo caso diventa un po’ complicato, perché superate le 3 variabili in ingresso il numero di riporti diventano 2 (o più, ma non in questo caso) di cui bisogna tenere conto quando si realizzano i sommatori per i bit successivi.

Se ci si gioca bene la proprietà modulare dei circuiti sommatori si ottiene un circuito il cui ritardo temporale è pari al logaritmo in base 2 del numero di bit della rappresentazione dei due numeri.

Quando si vanno a sommare/moltiplicare numeri in floating point diventa più complicata, poiché bisogna denormalizzare le mantisse (in modo da poterle allineare). Ciò è fattibile tramite multiplexer (poiché ci permettono di “spostare” le cifre) ma rimane comunque complicato, tant’è che ancora oggi ai dispositivi serve un coprocessore che si occupi soltanto di quelli.

Ora introduciamo la Unità Aritmetico Logica (ALU). È una parte importante del processore ed è rappresentata in genere con un simbolo particolare:



I bit di Controllo servono a decidere quale operazione eseguire sulle variabili di ingresso A e B. Si possono eseguire le operazioni algebriche (SOMMA, CAMBIO DI SEGNO, PRODOTTO) e quelle logiche (NOT, AND, OR, XOR). Ciò funziona grazie a dei multiplexer. La differenza tra i circuiti che abbiamo visto finora e l’utilizzo della ALU è che il numero di bit in uscita è uguale al numero di bit dei valori in ingresso. I valori in uscita, quindi, potrebbero non essere corretti (poiché magari servirebbe un 33° bit che non c’è) -> per capire quando si è in uno di questi casi la ALU emette dei fili ausiliari in cui inserisce eventuali riporti (o evidenzia una situazione di overflow). Sfruttando Questi fili ausiliari si può verificare se il risultato di un’operazione è uguale o diverso da 0.

Per fare A+B La ALU passa i valori in ingresso nel circuito sommatore. Per fare A-B fa la stessa cosa ma nega i bit di B (a cui viene sommato 000001). Quindi il circuito sommatore dell’ALU è un circuito sommatore normale in cui però i fili trasportanti i valori di b entrano in dei multiplexer che, se la loro variabile di controllo misura 1, restituiscono il valore ribaltato (e la variabile di controllo viene inviata all’ingresso del riporto del sommatore della prima cifra, così da avere il + 00000001 nel caso del ribaltamento di segno in complemento a 2).

Aggiungendo poi un’altra variabile di controllo e un AND prima di ogni full adder (che prende in ingresso una cifra di A e la seconda variabile di controllo) si possono avere altre due funzioni: la funzione identità (per B) e la funzione negazione (sempre per B) (il tutto senza usare un secondo multiplexer).