4 Rappresentazione di numeri relativi ($\mathbb{Z}$)

In molti sistemi elettronici è necessario rappresentare valori numerici interi relativi. L'informazione sul segno deve quindi far parte della rappresentazione stessa, come elemento specifico aggiuntivo oppure intervenendo sulla legge che associa valore e rappresentazione, attribuendo valore negativo a una parte delle rappresentazione stessa.

4.1 Rappresentazione in modulo e segno (MS)

Il modo più intuitivo di introdurre l'informazione del segno, usato nell'algebra classica, è quello di introdurre un elemento aggiuntivo da anteporre al valore assoluto del numero, rappresentato con la notazione posizionale già vista in precedenza. Solitamente si usa il simbolo + (opzionale) per valori non negativi oppure il simbolo − per i valori negativi.

In un sistema elettronico digitale, per evitare di introdurre un nuovo simbolo, si preferisce inserire l'informazione del segno nella stessa legge di rappresentazione, dividendo i valori rappresentabili in valori positivi e negativi sulla base di una cifra aggiuntiva, nella posizione più significativa, per la quale sono ammessi solo 2 valori.

Restringiamo nel seguito l'attenzione al caso $B = 2$, di maggior interesse pratico per i sistemi digitali; in questo caso la cifra aggiuntiva si adatta naturalmente alla rappresentazione del segno. La convenzione maggioritaria prevede che 0 indica i valori positivi mentre 1 i negativi. La cifra del segno precede il valore del modulo. Nel caso del modulo nullo sia hanno 2 rappresentazioni possibili: 00…00 (+0) e 10… 00 (−0).

4.1.1 MS: range di rappresentazione

Se abbiamo a disposizione complessivamente n cifre binarie e una la usiamo per il segno, per il modulo ne restano $(n - 1)$, per cui avremo per il modulo stesso un range di valori rappresentabili pari a $0 \leq |x| < 2^{n-1} - 1$. Per ogni valore non nullo del modulo si avranno 2 diversi valori opposti. Quindi in totale il range rappresentabile sarà $-(2^{n-1} - 1) \leq x \leq (2^{n-1} - 1)$.

Il totale dei valori numerici rappresentati è quindi pari a $N = (2^{n-1} - 1) + (2^{n-1} - 1) + 1 = 2^n - 1$. Rispetto al massimo valore possibile 2^n, si ha la perdita di un codice dovuta alla doppia rappresentazione dello 0.

4.1.2 MS: legge di rappresentazione

La legge di rappresentazione è la relazione che lega tra loro la rappresentazione al valore effettivo del numero da rappresentare. Ha interesse esprimere la legge di rappresentazione in forma diretta (dato il valore x, determinare la sua rappresentazione X) e in forma inversa (data la rappresentazione X, ricavare il valore rappresentato x). La quantità rappresentante X, nei sistemi elettronici binari, sarà in genere un comune valore binario a n bit e stabiliamo di chiamare X_i i singoli bit della rappresentazione.

Non è difficile scrivere la legge di rappresentazione nel caso di valori in modulo e segno su n bit. Si ha:

$$X = x \qquad \text{se } 0 \le x \le (2^{n-1} - 1)$$

$$X = 2^{n-1} + |x| = 2^{n-1} - x \qquad \text{se } -(2^{n-1} - 1) \le x \le 0 \qquad [4.1]$$

La legge di rappresentazione si può invertire abbastanza facilmente, ottenendo le formule [4.2].

$$x = X \qquad \text{se } X \le (2^{n-1} - 1)$$

$$x = -(X - 2^{n-1}) = 2^{n-1} - X \qquad \text{se } X \ge 2^{n-1} \qquad [4.2]$$

Poiché la condizione che permette di scegliere tra le due espressioni della [4.2] è legata esclusivamente al valore del bit Xn-1, possiamo riscrivere la legge inversa [4.2] esplicitando il valore dei singoli bit:

$$x = (1 - 2X_{n-1})(X_{n-2}2^{n-1} ... + X_1 2 + X_0) \qquad [4.3]$$

Appare evidente che il bit X_{n-1} del segno (MSB) agisce modificando il segno dei pesi associati a tutti gli altri bit.

4.1.3 MS: operazioni con un singolo operando

Per valutare concretamente le caratteristiche di una rappresentazione per numeri relativi, conviene individuare gli algoritmi che permettono di trovare, per le operazioni più comuni, la rappresentazione del risultato a partire dalle rappresentazioni degli operandi. Partiamo dalle operazioni che coinvolgono un singolo operando.

In particolare, data la rappresentazione X di x, esamineremo come si ottiene la rappresentazione del valore inverso $-x$ e del valore assoluto $|x|$. Esamineremo inoltre come si possa rappresentare x avendo a disposizione una cifra in più, cioè come passare da una rappresentazione con n a una con n+1 cifre[1] e le condizioni per eseguire il passaggio inverso[2].

Per indicare le cifre binare della rappresentazione X usiamo valori binari tra parentesi quadre, eventualmente con pedici numerati e usiamo la doppia freccia per indicare l'associazione tra valore e rappresentante: $x \leftrightarrow X = [X_{m-1}; X_{m-2}; ... ; X_1; X_0]$.

4.1.3.1 Cambio segno

Per questa rappresentazione, l'operazione di cambio segno è diretta e si ottiene complementando[3] il bit più significativo. La rappresentazione dell'inverso è $-x \leftrightarrow [\overline{X}_{n-1}; X_{n-2}; ... ; X_1; X_0]$.

4.1.3.2 Estrazione del valore assoluto

Anche l'estrazione del valore assoluto è diretta e si ottiene attribuendo forzatamente il valore 0 al bit del segno: $|x| \leftrightarrow [0; X_{n-2}; ... ; X_1; X_0]$.

4.1.3.3 Estensione e riduzione del numero delle cifre

Se desideriamo rappresentare in modulo e segno con $n + 1$ bit un valore di cui è nota la rappresentazione su n bit, occorre aggiungere uno 0 tra la cifra più significativa contenente il segno e la precedente. Allo stesso modo, una rappresentazione su $n + 1$ bit che presenta uno 0 come seconda cifra più significativa, può essere ridotta a n bit eliminando proprio lo 0.

---

1 Estensione della rappresentazione.
2 Ed eventualmente come si può determinare la rappresentazione ridotta di x.
3 Con l'operazione di complemento si intende qui $\overline{X}_i = 1 - X_i$.

4.1.4 MS: Operazioni con 2 operandi

4.1.4.1 Somma e differenza

Come avviene sempre per i numeri relativi, l'operazione di differenza $x - y$ si può sempre ricondurre a una operazione di somma con l'opposto del sottraendo $x + (-y)$. Avendo già risolto il problema del cambio di segno, possiamo quindi concentrarci esclusivamente sul problema della somma.

L'esecuzione della somma tra numeri relativi rappresentati in modulo e segno richiede un'analisi preliminare dei segni e la disponibilità, nel caso di valori discordi, di un operatore di confronto tra i moduli[4] e di un sottrattore.

Il bit di uscita *OV* (overflow) indica con il valore 1 che il risultato della somma non è rappresentabile, e questo può avvenire esclusivamente nel caso di operandi concordi in cui la somma dei moduli non è rappresentabile con $(n - 1)$ bit.

Nella tabella seguente è mostrata tutta la possibile casistica che occorre affrontare nell'implementazione della somma per numeri binari in modulo e segno. Si hanno a disposizione i risultati della somma dei moduli S_{xy} con il relativo riporto C e i risultati delle differenze tra i moduli D_{xy} con prestito B_{xy} e D_{yx} con prestito B_{yx}. Ovviamente B_{xy} e B_{yx} non potranno mai essere contemporaneamente 1 e nello studio della casistica è sufficiente l'uso di uno dei due, per esempio B_{xy} la cui presenza a 1 fornisce l'informazione $|x| < |y|$.

X_{n-1}	Y_{n-1}	B_{xy}	R_{n-1}	$[R_{n-2};\ldots;R_0]$	OV
0	0	-	0	S_{XY}	C
1	1	-	1	S_{XY}	C
0	1	0	0	D_{XY}	0
0	1	1	1	D_{YX}	0
1	0	0	1	D_{XY}	0
1	0	1	0	D_{YX}	0

Tabella con la casistica della somma in MS

4.1.5 Conclusioni

A conclusione dell'esame delle caratteristiche della rappresentazione modulo è segno, possiamo elencare alcuni vantaggi:

- intuitiva
- semplicità delle operazioni con un singolo operando
- semplicità nella gestione della moltiplicazione tra relativi, che viene facilmente ricondotta alla moltiplicazione tra interi assoluti più la gestione del segno

Per contro altri aspetti sembrano problematici:

- assenza di biunivocità tra rappresentante e rappresentato (lo 0 ha 2 rappresentazioni equivalenti, una con il segno positivo e una con il segno negativo)

4 Per confrontare i moduli $|x|$ e $|y|$ può essere usato un sottrattore $|x| - |y|$. Nel bit di prestito finale è contenuta l'informazione che il minuendo $|x|$ è strettamente minore del sottraendo $|y|$.

- l’esecuzione della somma richiede l’analisi dei segni e un hardware specifico in grado di eseguire il confronto tra valori assoluti e la somma oppure la sottrazione.

Proprio in seguito a queste limitazioni, la rappresentazione in modulo e segno non è particolarmente diffusa nei sistemi digitali in virgola fissa, in cui spesso sono preferite rappresentazioni in cui l'esecuzione della somma si ottiene in modo più diretto.

4.2 Complemento a 1 (C1)

Per superare le limitazioni presentate dalla MS, senza complicare troppo l’esecuzione delle operazioni elementari, si ricorre a rappresentazioni che contengono valore di originale, mantenendone il segno, ma sommando “qualcosa” per ricondurre la rappresentazione a un valore positivo. Visto che il valore minimo rappresentabile in MS è $-(2^{n-1} - 1)$, una prima idea è quella di sommare l’opposto di questa quantità ai valori negativi e contestualmente porre il bit più significativo a 1. In totale il valore che viene sommato ai valori negativi è quindi $(2^n - 1)$ Questa rappresentazione si definisce C1.

4.2.1 C1: legge di rappresentazione

Per il solito range definito da *n* bit, si ha quindi

$$X = x \qquad \text{se } 0 \le x \le (2^{n-1} - 1)$$

$$X = (2^n - 1) + x = (2^n - 1) - |x| \qquad \text{se } -(2^{n-1} - 1) \le x \le 0 \qquad [4.x]$$

La legge di rappresentazione si può invertire abbastanza facilmente, ottenendo le formule [4.x+1].

$$x = X \qquad \text{se } X \le (2^{n-1} - 1)$$

$$x = X - (2^n - 1) \qquad \text{se } X \ge 2^{n-1} \qquad [4.x+1]$$

Si nota come anche in C1 la rappresentazione dello 0 non è univoca e corrisponde alle 2 alternative [00...00] insieme a [11...11]. Se nelle [4.x+1] si evidenzia il ruolo del bit più significativo, che è 0 per la prima delle 2 equazioni e 1 per la seconda, si ottiene

$$x = X_{n-1}(2^{n-1} - 2^n + 1) + X_{n-2}2^{n-2} + \ldots + X_1 2^1 + X_0$$

$$= -X_{n-1}(2^{n-1} - 1) + X_{n-2}2^{n-2} + \ldots + X_1 2^1 + X_0 \qquad [4.x+1]$$

che somiglia alla normale rappresentazione binaria, solo che il bit più significativo ha peso negativo e pari in modulo al massimo valore positivo rappresentabile , cioè $-(2^{n-1} - 1)$. Ovviamente per i numeri positivi, in cui $X_{n-1} = 0$, il peso del bit più significativo non ha alcuna importanza.

4.2.2 C1: operazioni con un singolo operando

4.2.2.1 Cambio segno

Si ottiene invertendo il valore di tutti i bit della rappresentazione, compreso quello più significativo. L'operazione non può dare mai overflow.

La rappresentazione dell'inverso è quindi $-x \leftrightarrow [\overline{X}_{n-1}; \overline{X}_{n-2}; \ldots ; \overline{X}_1; \overline{X}_0]$.

La semplicità con cui si ottiene questa operazione è il principale punto di forza della rappresentazione C1.

4.2.2.2 Estrazione del valore assoluto

Per l'estrazione del valore assoluto occorre identificare il segno del valore rappresentato: nel caso di valore positivo non occorre fare nulla, altrimenti occorre applicare l'operazione di cambio segno vista nel paragrafo precedente[5]: $|x| \leftrightarrow$ if(X_{n-1} =0; [X_{n-1}; ... ; X_1; X_0]; [$\overline{X}_{n-1}$; $\overline{X}_{n-2}$; ... ; $\overline{X}_1$; $\overline{X}_0$]).

4.2.2.3 Estensione e riduzione del numero delle cifre

Si può estendere la rappresentazione di un numero replicando il bit più significativo, cioè Xn = Xn-1 e lasciando inalterati tutti i bit rimanenti.

Questa operazione è coerente con il peso dei bit nella rappresentazione C1, dove il bit più significativo ha peso negativo $-(2^{n-1} - 1)$. Infatti l'aumento del numero di bit corrisponde a modificare il peso del bit $(n-1)^{esimo}$ in 2^{n-1} e ad aggiungere un bit di peso $-(2^n - 1)$. Sommando i 2 pesi si ottiene $-(2^n - 1) + 2^{n-1} = -(2^{n-1} - 1)$.

Viceversa è possibile ridurre i bit della rappresentazione di un numero da $n + 1$ a n solo se i 2 bit più significativi sono concordi, eliminando quindi uno dei due.

4.2.2 C1: operazioni con 2 operandi

Somma e differenza

Per la differenza, vista la facilità dell'operazione di cambio segno, vale quanto già detto per MS. La somma viene facilitata dal fatto che i bit hanno sempre gli stessi peso e quindi bit corrispondenti possono essere sommati.

4.2.5 Conclusioni

A conclusione dell'esame delle caratteristiche della rappresentazione C1, si nota che, pur avendo contenuto la complessità delle più comuni operazioni a singolo operando, l'operazione di somma è stata semplificata, anche se ci sono margini per ulteriori progressi.

4.3 Complemento a 2 (C2)

Si ottiene un significativo miglioramento, restando nell'ambito delle rappresentazioni che permettono l'esecuzione diretta della somma, facendo in modo che il peso del bit più significativo sia esattamente una potenza di 2. La rappresentazione C2 è un esempio di questo approccio, con notevoli proprietà che discendono direttamente dalle proprietà dell'operazione modulo.

4.3.1 C2: legge di rappresentazione

Procedendo come per il C1, nel solito range di n bit, si ha

$$X = x \qquad \text{se } 0 \le x \le (2^{n-1} - 1)$$

$$X = 2^n + x = (2^n - 1) - |x| + 1 \qquad \text{se } -2^{n-1} \le x < 0 \qquad [4.x]$$

La legge di rappresentazione si può invertire abbastanza facilmente, ottenendo le formule [4.x+1].

$$x = X \qquad \text{se } X \le (2^{n-1} - 1)$$

5 Si è fatto ricorso alla stuttura di uso comune if(test; operazione se vero; operazione altrimenti).

$$x = X - 2^n \qquad \text{se } X \geq 2^{n-1} \qquad [4.x+1]$$

Si nota subito che in C2 la rappresentazione dello 0 è univoca, associata ai valori positivi e non anche ai negativi, come in MS e C1. La rappresentazione C2 permette quindi di rappresentare un valore in più, [10...00] corrisponde al minimo valore negativo -2^{n-1}. Anche in C2 si può evidenziare il ruolo del bit più significativo, ottenendo la semplice espressione:

$$x = -X_{n-1}2^{n-1} + X_{n-2}2^{n-2} + ... + X_1 2^1 + X_0 \qquad [4.x+1]$$

che conviene assumere come la normale forma di decodifica di una rappresentazione C2.

Cambio segno

CaSi ottiene invertendo il valore di tutti i bit della rappresentazione e sommando 1. L'operazione può dare overflow nel caso in cui il valore di partenza sia pari a -2m-1, in quanto il valore positivo di uguale valore assoluto non è rappresentabile.

Estensione e riduzione del numero delle cifre

Si può estendere la rappresentazione di un numero ponendo
bm-1=bm-2m
bm-2=0

bi=ai

b0=a0

Viceversa è possibile ridurre i bit della rappresentazione di un numero da m+1 a m solo se bm-2=0. In questo caso la rappresentazione con un bit in meno si ottiene eliminando semplicemente il bit nullo.

Somma

La somma tra numeri in C2 può essere eseguita semplicemente usando un sommatore a n+1 bit, dopo aver esteso di 1 bit le rappresentazioni degli addendi.

La facilità con cui si può gestire la somma ha decretato il successo di questa rappresentazione, che è quella maggiormente usata nelle architetture digitali che trattano numeri interi relativi.

4.3 Traslazione (T)

3.2.3

In alcune architetture (per esempio in alcuni circuiti di conversione da quantità digitale a tensione analogica, DAC) si fa uso della rappresentazione in traslazione, che si ottiene semplicemente facendo corrispondere il minimo valore da rappresentare allo 0 della rappresentazione e

4.2 Passaggio da una rappresentazione all'altra

Può capitare di dover interfacciare sottosistemi in cui valori numerici relativi siano rappresentati con strategie diverse. È importante disporre quindi di circuiti digitali in grado di trasformare le rappresentazioni da una tipologia all'altra, a parità di valore, segnalando il caso di eventuale overflow (possibile da C2 oppure T verso MS oppure C1).

Esaminiamo quindi le possibili soluzioni, tenendo presente che in alcuni casi conviene ottenere le trasformazioni desiderate componendone in cascata 2, usando un terzo formato come risultato intermedio. I vari sistemi digitali proposti fanno uso, come elemento base di elaborazione, del sommatore tra 2 bit, che genera in uscita il bit di somma e l'eventuale riporto, oppure fanno uso del sottrattore e in questo caso in uscita si ha il bit di differenza e l'eventuale prestito. In Figura X vengono mostrati i simboli che indicano queste funzionalità di base.

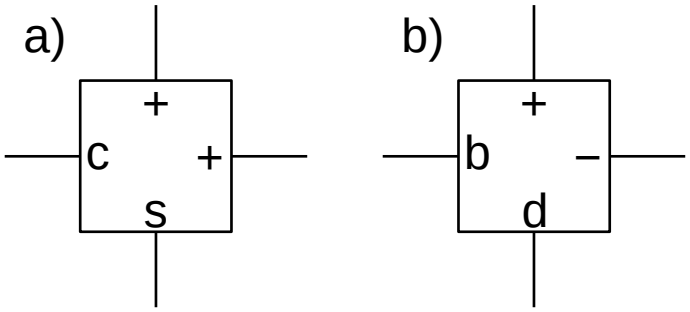


Figura X: Blocchi di elaborazione. a) Sommatore a 2 bit; in uscita "s" è il bit della somma e "c" il riporto verso il peso superiore. b) Sottrattore a 2 bit; "d" è il bit della differenza e "b" la richiesta di prestito dal peso superiore.

Si può osservare che il sommatore[6], quando uno dei due ingressi è 0, all'uscita "s" presenta l'altro ingresso inalterato poiché 0 è l'elemento neutro della somma. Invece se un ingresso è 1, la somma e l'altro ingresso sono necessariamente diversi, quindi se l'altro ingresso è 0 l'uscita è 1 e viceversa.

4.2.1 Da MS agli altri formati

Poiché la rappresentazione MS è quella generalmente usata nelle operazioni di calcolo manuale, e quindi spesso usata dai sottosistemi di input, è importante disporre dei circuiti digitali in grado di eseguire la conversione verso tutti gli altri formati, più utili per la rappresentazione interna nella macchina e per l'esecuzione delle operazioni.

Sulla base delle leggi di conversioni definite nei paragrafi precedenti, in Figura X viene mostrato il sistema digitale che converte da MS a C1, che sfrutta la proprietà di invertitore di bit del blocco sommatore. In questo caso l'ingresso di controllo è costituito dal segno: se il valore in ingresso è positivo, l'uscita è una replica dell'ingresso, altrimenti tutti i bit del modulo vengono invertiti.

6 Ma anche il sottrattore ha la stessa proprietà.

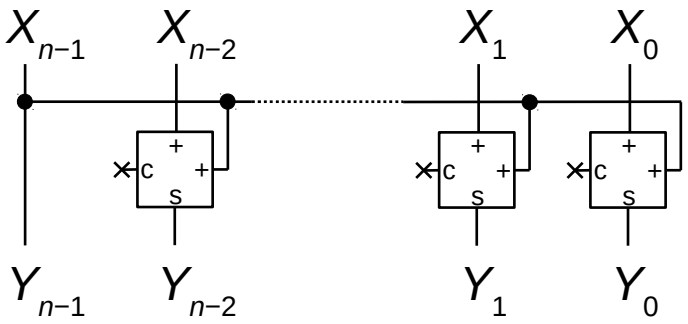


Figura X: Conversione MS→C1 e viceversa

È interessante notare che la rete in Figura X funziona anche nella direzione opposta, convertendo da C1 a MS. L'algoritmo di conversione infatti è reversibile e la doppia applicazione dell'operazione dell'invertitore riporta al dato di partenza. Per proseguire nella conversione verso il formato C2 occorre, per valori negativi, incrementare il valore espresso in formato C1. Quindi in Figura X troviamo questo ulteriore sottosistema da porre in cascata al precedente. Il bit più significativo, che mantiene l'informazione del segno, costituisce l'ingresso di una catena di sommatori in grado di incrementare il valore in ingresso. La conversione C1C2 non può dare overflow; si osservi che il caso Y = 11…11, proveniente dal valore MS -0 e unico caso in cui il sommatore in Figura X ha riporto in uscita, applicato alla rete dà in uscita il valore corretto Z = 0.

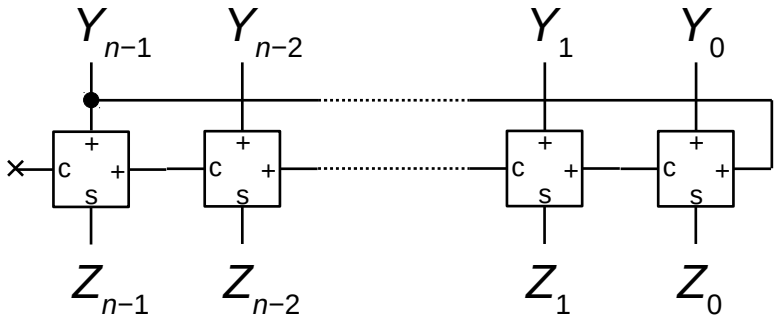


Figura X: Conversione C1→C2.

Per completare le trasformazioni possiamo aggiungere che la rappresentazione in traslazione (T) si ottiene immediatamente a partire dalla C2 invertendo il bit più significativo. In Figura X viene mostrato questo ulteriore blocco. Come nel caso delle conversioni MS>C1 e C1>MS, anche in questo caso la stessa rete può essere usata per ottenere la trasformazione inversa.

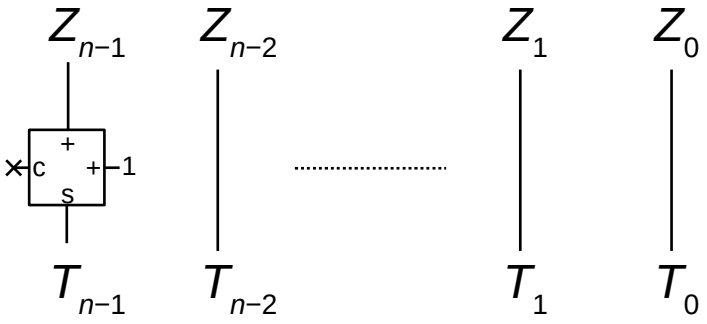
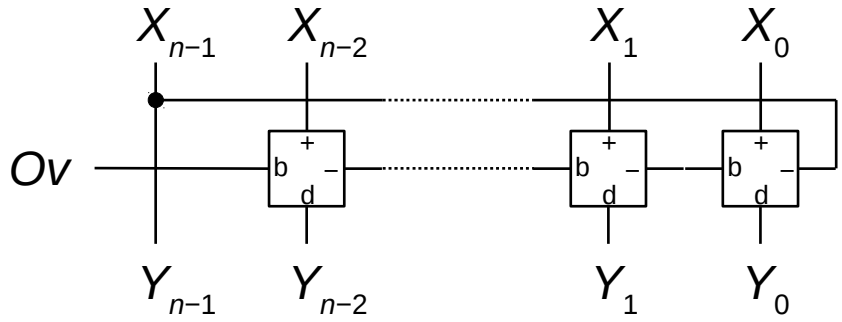


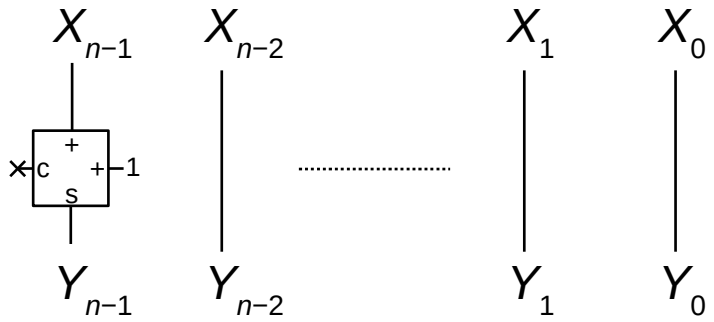
Figura X: Conversione C2→T e viceversa

4.2.2 Altre conversioni di formato

La proposta di soluzioni per convertire MS in C1, C2 e T ci ha permesso di prendere in esame come passaggi intermedi oppure come casi di trasformate valide nelle 2 direzioni, diverse altre conversioni. Se a queste aggiungiamo a queste la trasformazione C2 C1 abbiamo un set completo di operatori per ottenere qualsiasi tipo di conversione.



C2>C1



C2>T e viceversa