Appendice A

Sintesi diretta di frequenza digitale

A.1 Sommario

La generazione di portanti sinusoidali, una delle funzioni chiave dell'elettronica applicata alle telecomunicazioni, è stata realizzata con tecniche analogiche sino agli anni '70 quando l'avvento dei circuiti a larga scala d'integrazione ha aperto la strada a realizzazioni digitali con prestazioni decisamente superiori, esse permettono infatti di variare la frequenza in istantaneamente e con continuità di fase raggiungendo risoluzioni dell'ordine del μHz, il tutto con costi e dimensioni contenuti.

L'unico aspetto nel quale le tecniche analogiche sono superiori è l'estensione spettrale che raggiunge diversi GHz pertanto la norma di progetto è di utilizzare tecniche digitali laddove possibile ed integrarle con tecniche analogiche negli altri casi.

A.2 DDFS

La sintesi diretta di frequenza digitale si basa sullo schema in Figura(A.1) che viene indifferentemente denominato DDS[1], NCO[2] oppure DDFS, esso genera una funzione a frequenza arbitraria e non lineare, quale il seno o il coseno[3], nei tre seguenti passi:

1. genera una sequenza di valori dell'argomento variabile in funzione della frequenza che si desidera ottenere.

2. associa alla sequenza generata le corrispondenti ampiezze del seno o del coseno.

[1] Direct Digital Synthesizer
[2] Numerical Control Oscillator
[3] caratterizzate tuttavia da un argomento φ lineare nel tempo

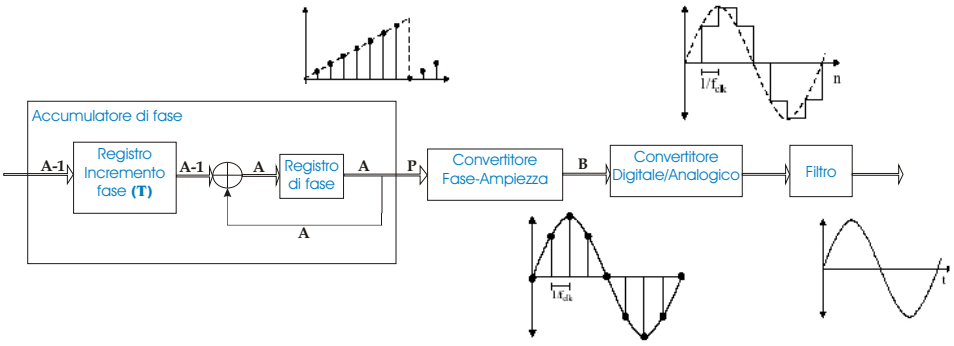


Figura A.1: Schema di principio DDFS

3. converte in analogico l'ampiezza espressa in digitale.

Con riferimento alla Figura(A.1) la sequenza dei valori dell'argomento, ciascuno rappresentato con A bit, viene generata mediante l'accumulatore di fase, esso si compone di un registro di fase e di un sommatore che aggiunge al valore precedentemente contenuto nel registro il valore T di una parola di sintonia contenuta in un registro ad A−1 bit[4]. Variando il valore T contenuto nel registro di incremento il progettista o l'operatore ha modo di variare la frequenza generata dal DDFS:

$$f_{out} = \frac{T \times f_{clk}}{2^A} \tag{A.1}$$

per variare la risoluzione in frequenza occorre invece cambiare il numero di bit A del registro di fase, si tratta di un valore da decidere in sede di progetto tenendo conto della applicazione cui è destinato il DDFS, valori elevati di A infatti aumentano la risoluzione, pari a $\frac{f_{clk}}{2^A}$, ma comportano una maggiore complessità circuitale e conseguentemente una minor frequenza di clock applicabile al sistema il ché per il teorema di Nyquist[5] si traduce in una riduzione della massima frequenza generabile.

Dopo aver generato la rampa di fase il convertitore fase/ampiezza la trasforma nella funzione sinusoidale prescelta, esso ha in ingresso P bits con $P \ll A$ in quanto l'utilizzo di tutti i bit che provengono dall'accumulatore di fase determina una complessità non gestibile indipendentemente dal modo in cui il convertitore è realizzato. Il troncamento della fase determina in uscita dal DDFS la presenza di spurie la cui distribuzione ed ampiezza può tuttavia essere controllata mediante una opportuna pianificazione delle frequenze.

[4] per ottemperare al criterio di Nyquist
[5] Appendice(B.2)

La sequenza discreta ottenuta dal convertitore fase-ampiezza viene applicata ad un DAC il cui numero di bit B in ingresso determina il massimo livello delle spurie da esso prodotte, l'ordine zero del campionamento da poi luogo ad una sagomatura $\frac{\sin(x)}{x}$ dello spettro che richiede una precompensazione. Il filtro anti-aliasing ha il compito di lasciare passare in uscita il solo

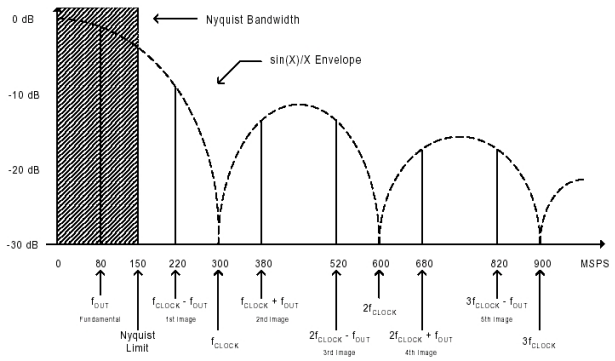


Figura A.2: Spettro in uscita dal DAC

spettro compreso nella banda $0 < f_{out} < \frac{f_{clk}}{2}$ detta di Nyquist.

A.3 Pianificazione delle frequenze

A.3.1 Spurie dovute al DAC

Come evidenziato in Figura(A.2) la non-linearità del DAC genera alla sua uscita oltre alla frequenza fondamentale f_{out} anche delle immagini aventi frequenza $f_{clk} \pm f_{out}$, scegliendo $f_{out} > f_{Nyquist}$ abbiamo che la prima immagine cade nella banda di Nyquist e quindi non può essere filtrata pertanto, tenendo conto anche della banda di transizione non nulla del filtro anti-aliasing, si ha che la massima frequenza generabile è dell'ordine di $0,4 f_{out}$.

La distribuzione delle spurie dovute al DAC è imposta dalla relazione tra f_{clk} e f_{out} mentre la loro ampiezza è determinata dal numero di bit B in particolare abbiamo che il rapporto $\frac{S}{N}$ espresso in dB tra il segnale ed il rumore di quantizzazione [10] vale:

$$\frac{S}{N}(dB) = 6.02 \times B + 1.76 + 20 \log_{10}(FFS) \tag{A.2}$$

dove con FFS si indica la percentuale del fondoscala a cui lavora il DAC.

La potenza del rumore di quantizzazione è costante per un dato valore di B pertanto se si riduce la banda utile ossia si lavora in una frazione della banda di Nyquist si ottiene un miglioramento del rapporto $\frac{S}{N}$.

A.3.2 Spurie dovute al troncamento della fase

La risoluzione in frequenza aumenta al crescere del numero di bit A nell'accumulatore di fase tuttavia, allo stato attuale della tecnica, soltanto i P bit più significativi possono essere applicati al convertitore fase-ampiezza che altrimenti diviene irrealizzabile.

Il troncamento da luogo ad un errore che è periodico e pertanto produce delle frequenze spurie in uscita dal DDFS, la loro ampiezza varia da un minimo di 0 quando i P bit eliminati sono tutti 0, condizione che si verifica per[6]:

$$GCD\left(T, 2^{(A-P)}\right) = 2^{(A-P)} \tag{A.3}$$

ad un massimo di $-6.02P$ dBc nel caso l'MSB[7] dei P bit troncati sia 1 e tutti gli altri siano 0 il che accade se:

$$GCD\left(T, 2^{(A-P)}\right) = 2^{(A-P)} \tag{A.4}$$

Per comprendere la distribuzione di queste spurie si parte dall'osservazione che il segnale d'errore è proprio la parola troncata espressa su R = A−P bits, la parola di sintonia ad essa associata è $ETW = T \bmod \left(2^R\right)$ la quale accumulandosi da luogo alla forma d'onda a dente di sega di Figura(A.3) avente frequenza fondamentale $F_0 = F_{clk} \times \left(\frac{ETW}{2^R}\right)$ e spettro rappresentato

Figura A.3: Andamento nel tempo della parola troncata

in Figura(A.4). Le righe spettrali che cadono in multipli dispari della banda

Figura A.4: Spettro dente di sega

di Nyquist hanno delle immagini proprio all'interno di essa e pertanto non sono eliminabili mediante filtraggio, si può soltanto ridistribuirne la potenza

[6] GDC indica il massimo comun divisore tra i due argomenti
[7] Most Significant Bit

mediante la tecnica del dithering che consiste nel distruggere la periodicità aggiungendo delle quantità randomiche prima del troncamento.

A.4 Convertitore Fase-Frequenza

Vi sono due tipi di convertitore fase-ampiezza:

ROM le fasi troncate in uscita dall'accumulatore di fase individuano delle locazioni di memoria in una ROM le quali contengono la corrispondente ampiezza del seno o del coseno. La simmetria del coseno e del seno consente di ridurre le dimensioni della ROM limitandosi a memorizzare i valori delle ampiezze per le sole fasi comprese tra 0 e 90° ed utilizzando i due bit più significativi della parola di fase troncata per ricostruire l'ampiezza corretta come in Figura(A.5).

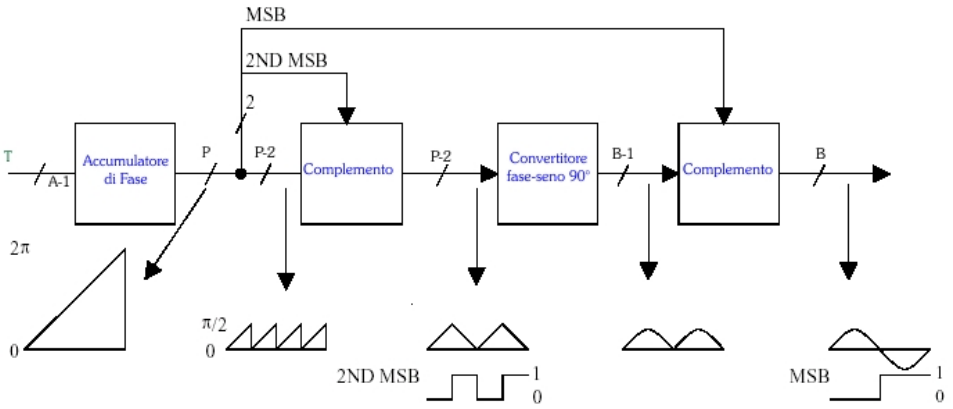


Figura A.5: Ottimizzazione del convertitore fase-ampiezza

CORDIC la simmetria del coseno e del seno viene utilizzata anche nella soluzione CORDIC [9], della quale esistono due forme denominate Vectoring CORDIC e Rotation CORDIC, quest'ultima è quella utile per la generazione di seno e coseno a partire dall'argomento, si cerca infatti di annullare l'angolo φ iniziale sommando e sottraendo ad esso degli angoli via via più piccoli, al crescere del numero di iterazioni aumenta la precisione del risultato e conseguentemente migliora l'approssimazione sia per il seno che per il coseno dell'angolo di partenza. L'algoritmo è adatto ad applicazioni digitali in quanto le rotazioni si basano su moltiplicazioni per potenze di due.

La ROM risulta una soluzione conveniente nel caso si scelga una fase troncata con al massimo 14 bit [14], oltre conviene la soluzione CORDIC la quale ha anche il vantaggio di produrre congiuntamente ed in maniera

intrinseca sia seno che coseno risultando ottimale per la realizzazione di un modulatore con portanti in quadratura, per questi motivi nel seguito verrà tralasciata la trattazione del DDFS basato sulla ROM.

A.4.1 Algoritmo CORDIC

Si basa su di una rotazione planare che trasforma un vettore (X_i, Y_i) in un nuovo vettore (X_n, Y_n), la forma matriciale di una generica rotazione

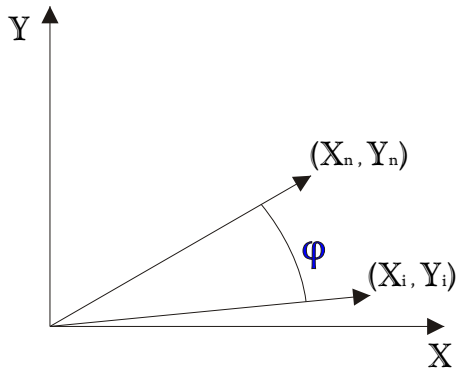


Figura A.6: Rotazione planare

planare di un angolo φ è

$$\begin{bmatrix} X_n \\ Y_n \end{bmatrix} = \begin{bmatrix} \cos\varphi & \sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \end{bmatrix} \tag{A.5}$$

trattandosi di un algoritmo iterativo la rotazione φ viene scomposta in n rotazioni φ_i ciascuna delle quali è espressa dalla

$$\begin{bmatrix} X_{i+1} \\ Y_{i+1} \end{bmatrix} = \begin{bmatrix} \cos\varphi_i & \sin\varphi_i \\ \sin\varphi_i & \cos\varphi_i \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \end{bmatrix} \tag{A.6}$$

da cui raccogliendo $\cos(\varphi_i)$ si ha

$$\boxed{\begin{bmatrix} X_{i+1} \\ Y_{i+1} \end{bmatrix} = \cos\varphi_i \begin{bmatrix} 1 & -\tan\varphi_i \\ \tan\varphi_i & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \end{bmatrix}} \tag{A.7}$$

quest'ultima forma è vantaggiosa in quanto il numero di moltiplicazioni passa da 4 a 3, l'ulteriore vincolo

$$\boxed{\tan\varphi_i = S_i 2^{(-i)}} \tag{A.8}$$

con $S_i \in \{-1; +1\}$ consente di ottimizzare l'implementazione su sistemi digitali in quanto le moltiplicazioni per potenze di due si ottengono facendo scorrere il contenuto di un registro in una direzione o nell'altra.

Il valore di S_i alla i_{esima} iterazione viene determinato richiedendo che sia minima la differenza tra l'angolo iniziale e l'angolo che si ottiene dalla somma degli angoli delle iterazioni precedenti, se tale differenza è minore di zero allora $S_i = -1$ altrimenti $S_i = +1$.

Il vincolo descritto dall'equazione(A.8) è fondamentale nella semplificazione dell'algoritmo in quanto $\cos(\varphi_i) = \cos(-\varphi_i)$ e quindi per una data iterazione n la produttoria dei $\cos(\varphi_i)$ delle iterazioni precedenti è nota a priori e vale

$$K_n = \prod_{i=0}^{n-1} K_i = \prod_{i=0}^{n-1} cos\left(\arctan\left(\frac{1}{2^i}\right)\right) = \prod_{i=0}^{n-1} \sqrt{1 + 2^{(-2i)}} \tag{A.9}$$

tale valore approssima $0,607252935008881$ per $n \to \infty$ ma già per n = 5 vale $0,607351770141296$ pertanto per algoritmi CORDIC con un numero di iterazioni maggiori di 10 si può considerare un numero costante soprattutto in considerazione della lunghezza finita dei registri. Il reciproco di K vale approssimativamente $1,64676025812107$ ed è denominato guadagno del CORDIC, se occorre che il vettore risultante abbia modulo unitario deve essere compensato utilizzando una delle seguenti strategie:

1. per la prima iterazione si ha $X_i = 1$, $Y_i = 0$, alla fine occorre dividere sia X_n che Y_n per il guadagno.

2. iniziando con $X_i = 0.607351770141296$, $Y_i = 0$ non è poi necessario dividere per il guadagno tuttavia le prestazioni sono inferiori rispetto alla soluzione precedente in quanto $0,607351770141296$ non è potenza di due e quindi la sua implementazione in un registro implica un errore iniziale.

3. compensazione del guadagno ad ogni iterazione.

Omettendo K per semplicità di rappresentazione si ha che l'i_{esima} iterazione del CORDIC è descritta dalle equazioni:

$$\begin{cases} X_{i+1} = & X_i - S_i 2^{(-2i)} Y_i \\ Y_{i+1} = & Y_i + S_i 2^{(-2i)} X_i \\ \varphi_{i+1} = & \varphi_i - S_i \arctan 2^{(-2i)} \end{cases} \tag{A.10}$$

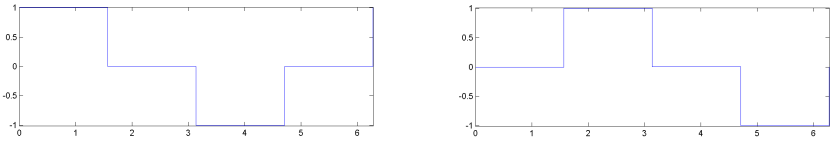
A.5 Descrizione modello matematico

Lo script Matlab che descrive l'NCO è ***NCO_Q.m*** *(Listato E.1.1)* , tramite ***Imposta_frequenze_Q*** *(Listato E.1.2)* vengono impostate la frequen-

za di clock e quella desiderata in uscita mentre ***Imposta_NCO_Q.m*** *(Listato E.1.3)* consente di impostare i parametri:

n_bit_acc	N° di bit accumulatore di fase
n_periods	N° di periodi della funzione generata
n_bit_cordic	N° di bit per la fase troncata
n_it_cordic	N° di iterazioni del CORDIC
n_bit_dac	N° di bit per seno e coseno prodotti dal CORDIC

L'accumulatore di fase in ***Crea_super_accumulatore_Q.m*** *(Listato E.1.4)* è modellato con un vettore contenente tutti i valori consecutivi della fase da applicare al CORDIC, in tale vettore sono memorizzate *n_periods* rampe di fase. ***Tronca_Q.m*** *(Listato E.1.5)* consente di passare da una rappresentazione della fase su *n_bit_acc* bits ad una su *n_bit_cordic* bits, tale vettore quantizzato è poi utilizzato da ***Crea_coseno_e_seno_Q.m*** *(Listato E.1.6)* il quale per ogni campione di fase invoca ***Forward_cordic_Q.m*** *(Listato E.1.7)* che opera riportando tutte le fasi nel primo quadrante in accordo alla Figura(A.5) ed eseguendo iterativamente per *n_it_cordic* volte le equazioni(A.10). Partendo con il coseno che vale $0,607252936517011$ invece che 1 non è necessario dividere dopo l'ultima iterazione per il guadagno del CORDIC[8] tuttavia occorre riportare i valori calcolati nel giusto quadrante, del seno e coseno ottenuti viene visualizzato un periodo nel dominio del tempo come in Figura(A.7) relativa al caso di $f_{clk} = 4 \times f_{out}$.



(a) Coseno (b) Seno

Figura A.7: Uscita DDFS nel tempo

Per la determinazione dello spettro viene utilizzato il metodo Multi-Taper in ***Visualizza_spettro_e_SFDR_p.m*** *(Listato E.1.9)* il quale mostra anche l'SFDR[9] determinato in ***Calcola_SFDR.m*** *(Listato E.1.10)* quale differenza tra il massimo assoluto dello spettro ed il maggior massimo relativo, un esempio dei risultati ottenuti è in Figura(A.8).

A.6 Descrizione modello VHDL

Il file ***NCO.vhd*** *(Listato F.1.1)* descrive lo schematico di Figura(A.9), quando

[8] Appendice(A.9)
[9] Spurious Free Dynamic Range

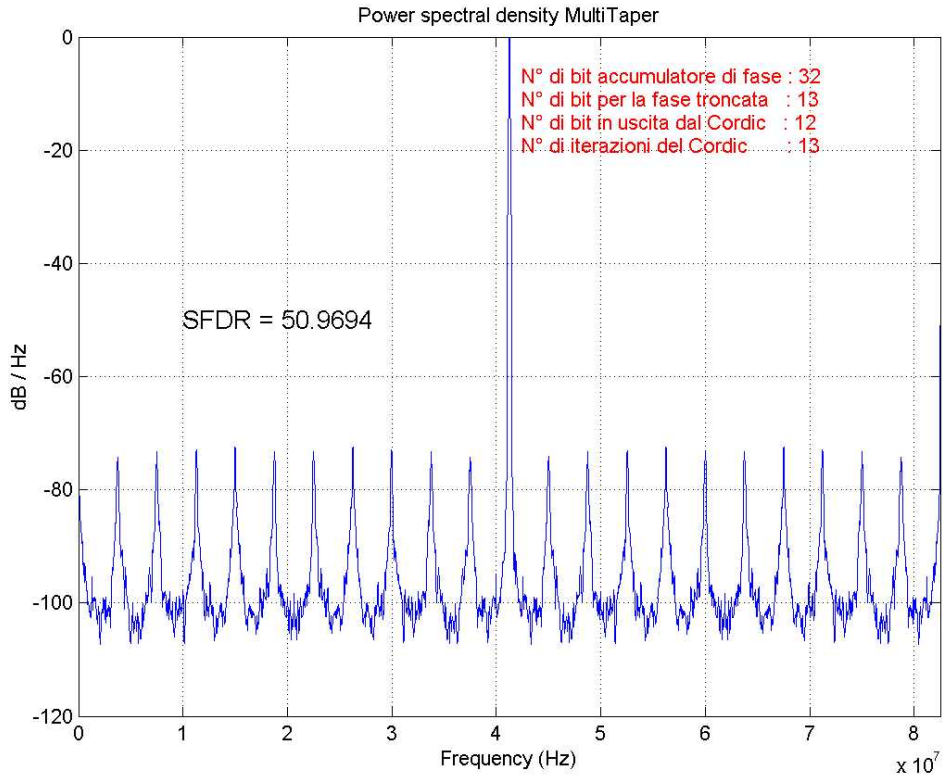


Figura A.8: Spettro Coseno Matlab

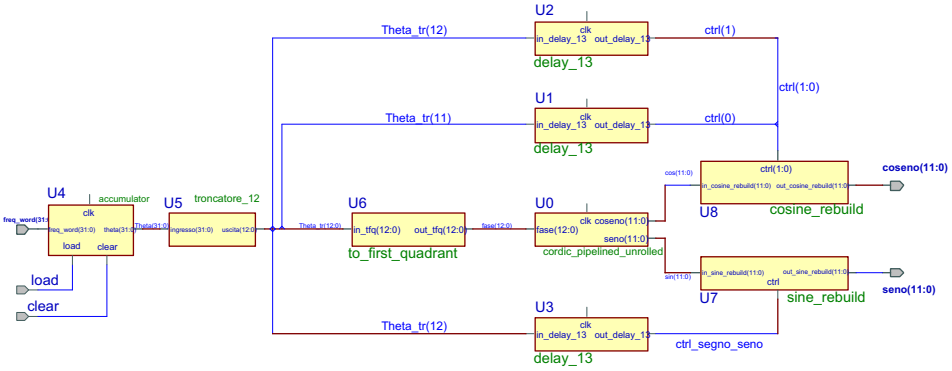


Figura A.9: NCO VHDL

LOAD è alto l'accumulatore ***Accumulator.vhd*** *(Listato F.1.2)* viene caricato con la parola di sintonia il cui valore in binario viene calcolato tramite lo script Matlab ***Calcola_incremento_fase_NCO.m*** *(Listato E.1.11)* ad ogni colpo di clock questo incremento si somma al valore precedentemente

contenuto nel registro a 32 bit per poi essere troncato a 12 bit tramite ***Troncatore_12.vhd*** *(Listato F.1.14)* , i 2 bit più significativi vengono ritardati da ***Delay_13.vhd*** *(Listato F.1.11)* di 13 periodi di clock in modo da poter associare al giusto quadrante il seno ed il coseno calcolati per un dato valore della fase che viene confinata tra 0° e 90° da ***To_first_quadrant.vhd*** *(Listato F.1.13)* , in pratica questi due bit vanno a pilotare delle inversioni in complemento a due realizzate condizionatamente in ***Cosine_rebuild.vhd*** *(Listato F.1.10)* e ***Sine_rebuild.vhd*** *(Listato F.1.12)* .

Coseno e seno vengono calcolati mediante un'architettura CORDIC non iterativa al fine di ottenere la massima velocità di clock di sistema, ogni iterazione è effettuata tramite ***Cordic_base_j.vhd*** *(Listato F.1.4)* in accordo alle equazioni(A.10) visivamente descritte in Figura(A.10).

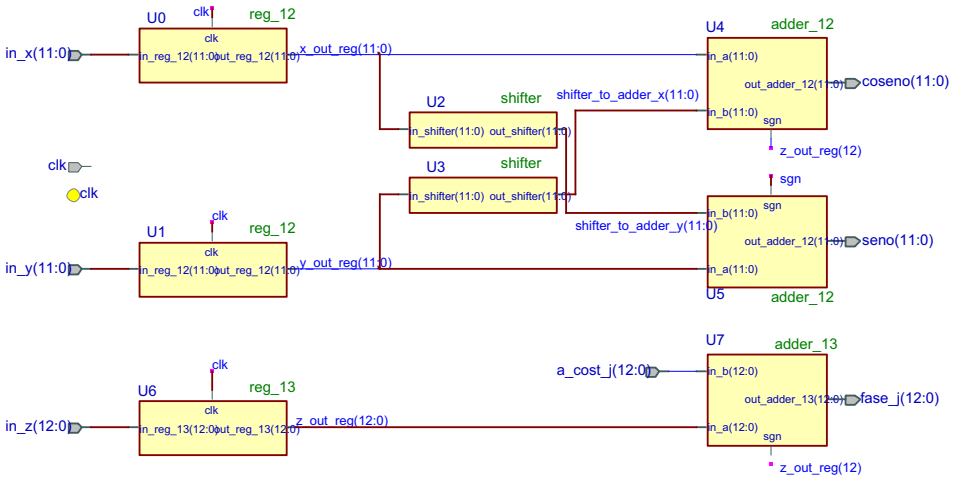


Figura A.10: Iterazione i_{esima}

Vengono eseguite 13 iterazioni in cascata, tramite la descrizione ***Cordic_pipelined_unrolled.vhd*** *(Listato F.1.3)* cui è associato lo schema in Figura(A.11) in esso i valori degli angoli elementari da sommare o sottrarre vengono calcolati ed espressi in binario tramite lo script Matlab ***Calcola_rotazioni_Cordic.m*** *(Listato E.1.8)* i cui risultati sono visualizzati in Tabella(A.1) che ben evidenzia come l'entità della rotazione i_esima si dimezza rispetto alla rotazione precedente.

Tramite lo script Matlab ***Visualizza_spettro_NCO_VHDL.m*** *(Listato E.1.12)* è possibile ottenere lo spettro del coseno generato dal CORDIC VHDL, un esempio di risultato è riportato in Figura(A.12).

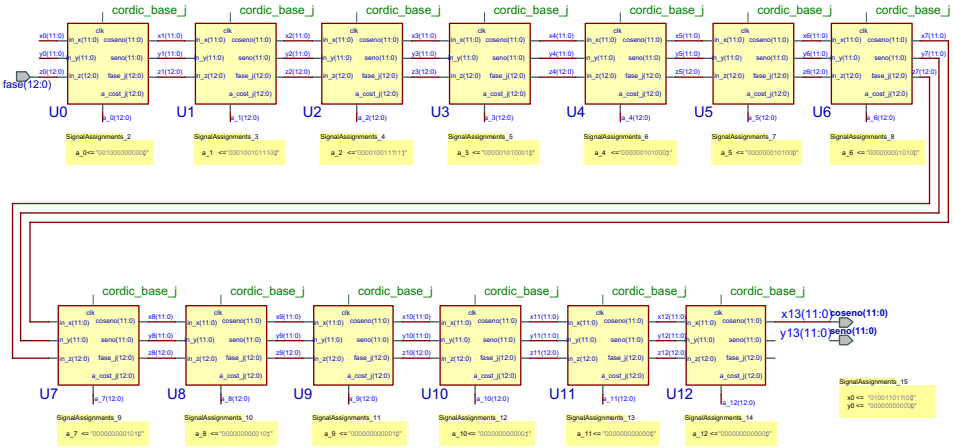


Figura A.11: Cordic Pipelined Unrolled

Iterazione	Angolo φ_i
1	$45,0000$
2	$26,5651$
3	$14,0362$
4	$7,1250$
5	$3,5763$
6	$1,7899$
7	$0,8952$
8	$0,4476$
9	$0,2238$
10	$0,1119$
11	$0,0560$
12	$0,0280$
13	$0,0140$

Tabella A.1: Rotazioni elementari CORDIC

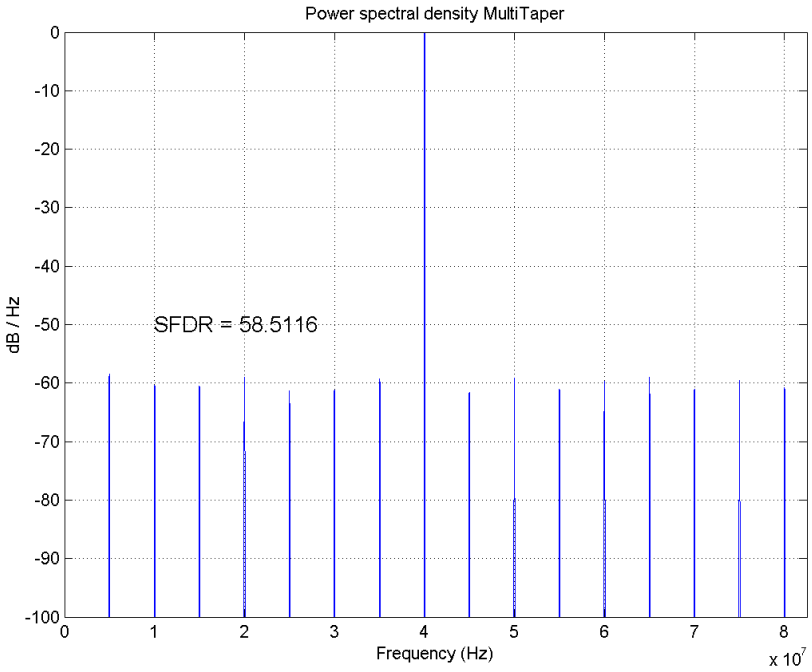


Figura A.12: Spettro coseno generato dal DDFS VHDL