Generatore di sequenze pseudo-casuali (Pseudo Noise Sequence Generator)

Giugno 2011

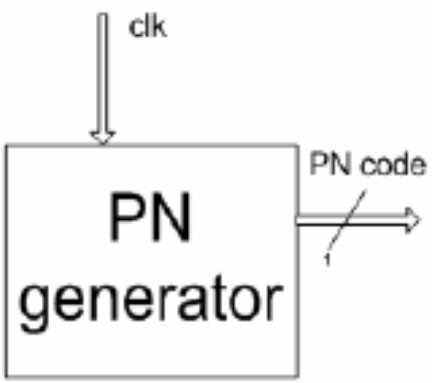
Enrico Molinari

Progetto realizzato per il corso di "Progettazione di Sistemi Microelettronici" (corso di Laurea Magistrale in Ingegneria Elettronica), con il supporto del Prof. Luca Fanucci, Università di Pisa, Facoltà di Ingegneria

Obiettivo: progettare un generatore di sequenze pseudocasuali per trasmissioni CDMA (IS-95 generatore ramo in fase) compatibile con le seguenti specifiche:

1) 15 stage PN generator
2) polimonio caratteristico: $x^{15} + x^{13} + x^9 + x^8 + x^7 + x^5 + 1$

Schema a blocchi:



La relazione finale del progetto deve contenere:

- Introduzione (descrizione algoritmo, possibili applicazioni, possibili architetture, etc.)
- Descrizione dell'architettura selezionata per la realizzazione (diagramma a blocchi, ingressi/uscite, etc.)
- Codice VHDL (con commenti dettagliati)
- Testbench per la verifica
- Conclusioni

Concetti di base ed applicazioni

Un generatore di sequenze pseudo-casuali PNSG (Pseudo Noise Sequence Generator), spesso chiamato anche generatore di numeri pseudo-casuali (PRNG, Pseudo Random Number Generator), è un dispositivo elettronico che, data in ingresso una particolare chiave ("key", spesso detta anche "seme", la cui introduzione impone lo stato iniziale del generatore, che è una macchina a stati finiti), fornisce in uscita una sequenza, un flusso, di numeri chiamata "keystream", ossia una chiave (un codice) di lunghezza arbitraria, generalmente molto più lunga, in termini di cifre, della chiave generatrice (il seme). I numeri costituenti la keystream, agli occhi di un osservatore esterno, risultano del tutto casuali, specialmente se questo non dispone di avanzate capacità computazionali; l'osservatore non può prevedere l'entità del numero (del simbolo, del bit) n-esimo, in procinto di essere prodotto e fornito in uscita, basandosi sulla semplice osservazione degli n-1 numeri precedentemente prodotti dal dispositivo e sul "trend" che ne ha caratterizzato la generazione. Generare sequenze di numeri realmente casuali è molto dispendioso da un punto di vista temporale, pertanto solitamente si opta per l'utilizzo di sequenze numeriche che sembrano casuali ma in realtà sono generate in modo deterministico. Le sequenze di numeri pseudo-casuali si avvicinano molto, in termini di caratteristiche e proprietà, a quelle realmente randomiche. Il seme e l'unico elemento realmente casuale, e la sequenza numerica realizzata dal generatore di numeri pseudo-casuali non è altro che un'espansione del seme in una stringa più lunga, mediante un algoritmo, ossia un processo deterministico, implementato da una particolare topologia circuitale. L'output è prodotto pertanto da una specifica funzione logica, e da ciò si deduce che l'introduzione dello stesso seme nel PNSG genererà sempre la stessa sequenza di output, e che la stringa pseudo-casuale non potrà essere infinita. La massima lunghezza della stringa di output, prima che questa cominci a ripetersi, è definita "periodo" della sequenza pseudo-casuale, solitamente si misura in bit oppure in cicli (periodi) di clock e dipende dalla particolare circuiteria implementata nel generatore e dal seme stesso. Se il seme introdotto ha lunghezza, misurata in bit, pari ad n, allora il periodo massimo della sequenza di output sarà pari a $(2^n - 1)n$ bit, e tale sequenza si ottiene attendendo un numero di periodi di clock pari a $2^n - 1$, dato che in un periodo di clock lo PNSG fornisce, in parallelo, una stringa di n bit.
La qualità di un generico PNSG è direttamente proporzionale al grado di pseudo-aleatorietà che caratterizza la generazione dei bit della sequenza di output e può essere rappresentata dal "coefficiente di sbilanciamento statistico" s, definito come:

$$s = \frac{1}{2^n - 1}$$

dove n rappresenta, come vedremo, il numero di stadi (di "stage") di cui consta l'architettura elettronica del generatore, ossia il numero di bit del seme con cui dobbiamo caricare lo stato iniziale del dispositivo. Supponiamo che l'uscita del generatore, ossia la sequenza pseudo-casuale, costituisca il segnale digitale di ingresso di un codificatore 2-PAM che associa al livello logico "0" il simbolo "1" e al livello logico "1" il simbolo "-1". La probabilità che il bit m-esimo della sequenza pseudo-casuale, codificata dai simboli appartenenti alla costellazione 2-PAM, risulti codificato dai simboli +1 o -1, ovvero che il bit stesso, prima della codifica, assuma valore 0 o 1, è data dalle seguenti espressioni:

$$Pr\{b_m = +1\} = \frac{1}{2}(1 - s) \qquad Pr\{b_m = -1\} = \frac{1}{2}(1 + s)$$

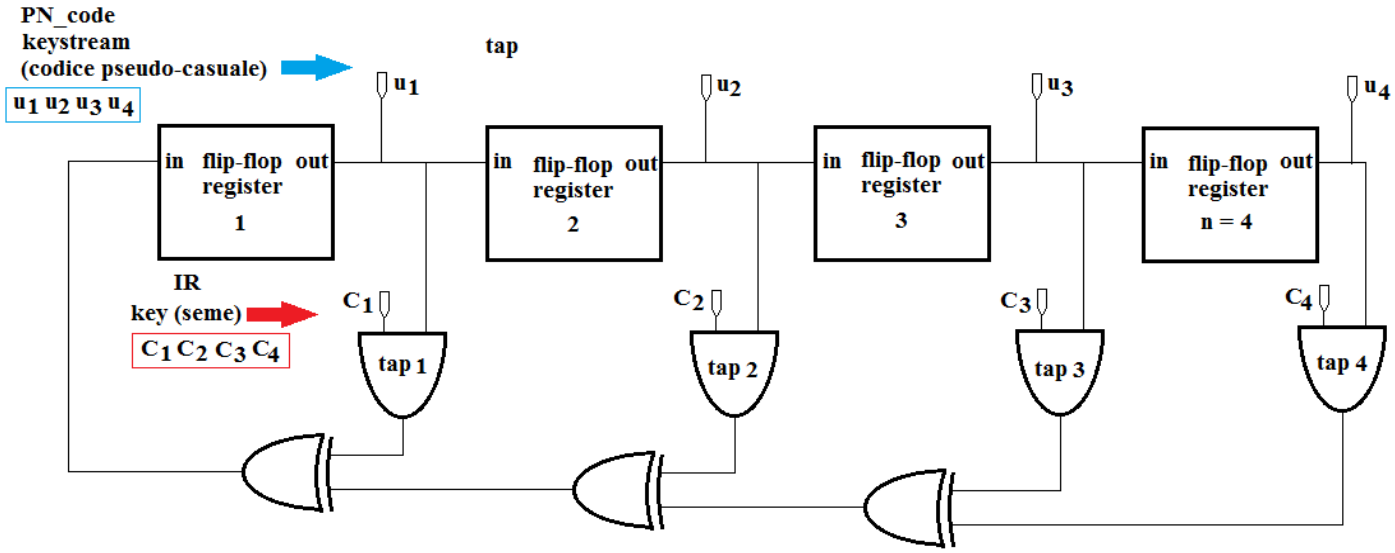
Evidentemente maggiore è n, ossia maggiore è la complessità dello PNSG, e minore è il valore di s, che tende a zero per valori di n molto alti, ossia per PNSGs realizzati mediante molti stadi; più s

tende a zero e più i due valori di probabilità suddetti risultato pari a ½, ossia più i vari stati nei quali il generatore può trovarsi sono equiprobabili. Pertanto maggiore è il numero n degli stadi di cui il generatore costa e più il funzionamento dello stesso si avvicina a quello di uno PNSG ideale (s = 0, "algoritmo" di produzione dei bit di uscita non più deterministico, più simile al processo stocastico di lancio di una moneta), il quale funzionerebbe, in linea teorica, per mezzo di un processo di generazione privo di memoria, durante il quale l'osservazione del suo stato, in un dato istante, non consente alcuna previsione sul futuro.
I dispositivi PNSG rappresentano una categoria di strumenti che continuano ad acquisire sempre maggiore rilevanza nello sviluppo di nuove tecnologie; la necessità di sfruttare sequenze di numeri pseudo-random si manifesta ormai in molteplici applicazioni, fra le quali:

- Crittografia (generazione di chiavi)
- Geometriche (metodo Monte Carlo)
- Modellazione di andamenti finanziari (azioni in Borsa)
- Informatica (addestramento reti neurali, inizializzazione algoritmi)
- Telecomunicazioni (trasmissione "Spread Spectrum", di cui parleremo)
- Elettronica (modellazione dispositivi in frequenza)
- Divertimento (videogiochi, giochi d'azzardo)

Possibili architetture di uno PNSG

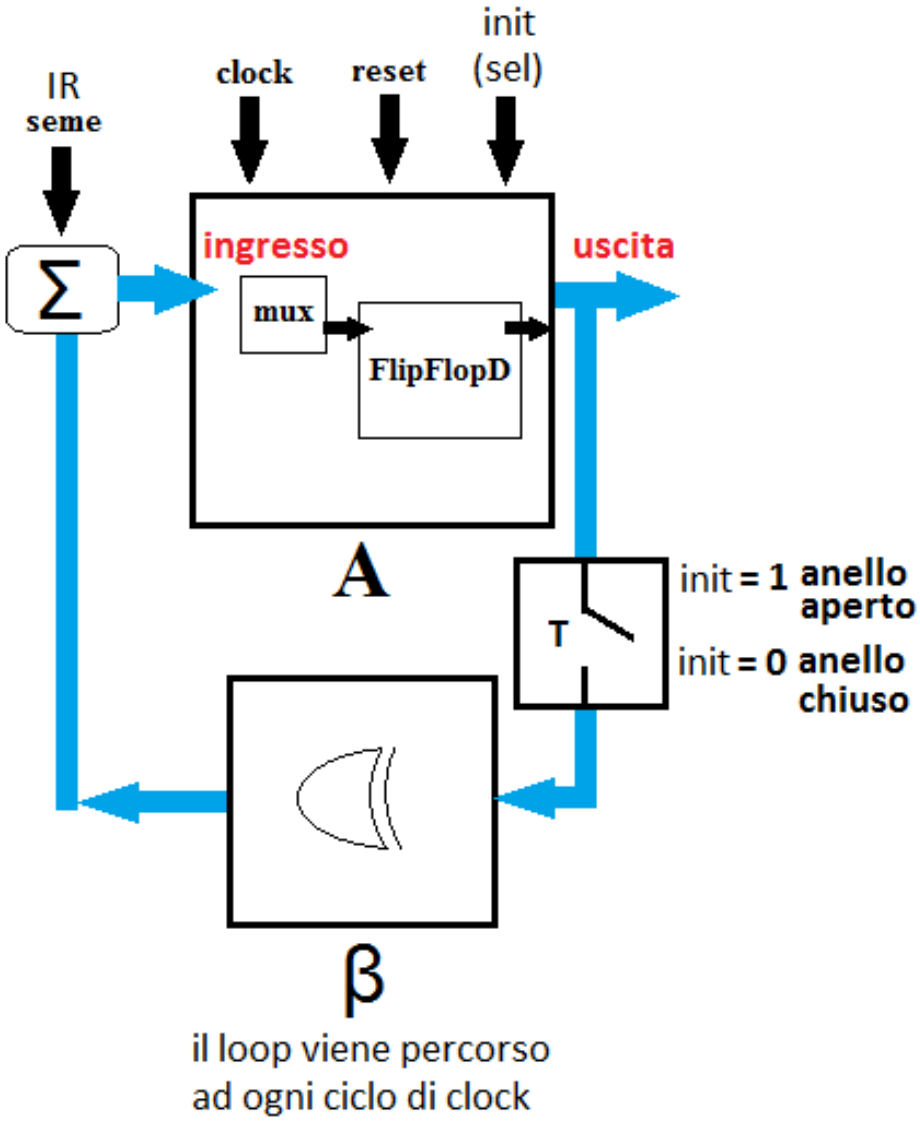
Una delle architetture maggiormente utilizzate, specialmente nel campo della crittografia, è quella basata sull'implementazione circuitale di un dispositivo noto come "registro a scorrimento lineare retroazionato" (LFSR, "Linear Feedback Shift Register"), di cui riportiamo una rappresentazione schematica di principio:

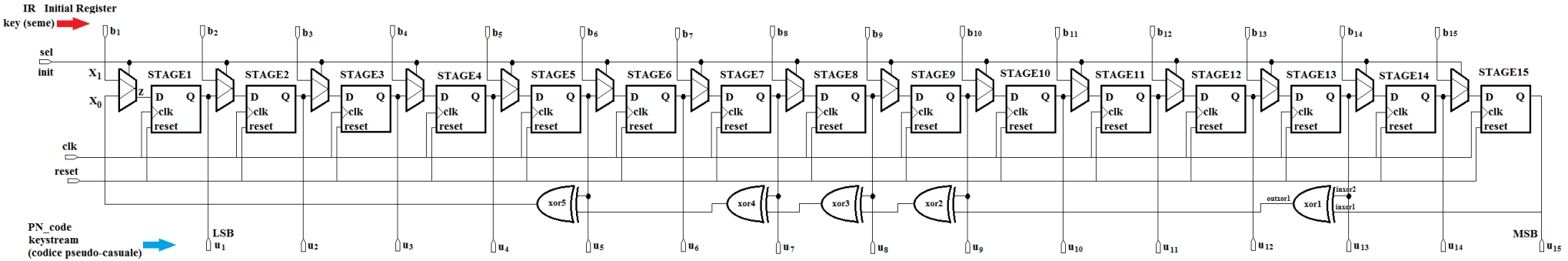


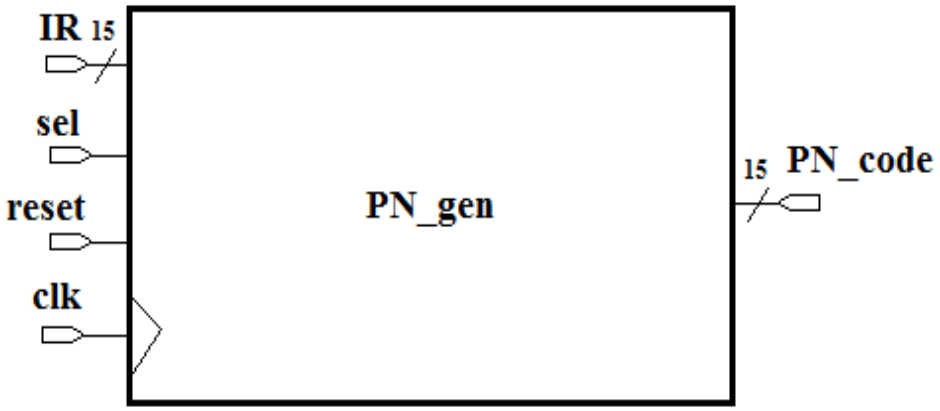
I bit che costituiscono il seme (C_1, …, C_n) vengono inseriti parallelamente al fine di settare lo stato iniziale del dispositivo; l'inserimento della key deve essere effettuato prima dell'avvio del segnale di clock, il quale è comune a tutti i registri (cioè gli stadi) disposti in cascata. Prima dell'avvio del clock, ossia prima che si verifichi il primo fronte di salita del clock (assumiamo di avere registri flip-flop positive edge triggered), l'uscita di ciascun registro seriale deve aver assunto stabilmente il valore logico "1". A partire dal primo fronte di salita del clock in poi, cioè dall'istante in cui, per la

prima volta, ciascun registro aggiorna il valore della propria uscita con quello del proprio ingresso, lo stato del dispositivo evolve nel tempo: il LFSR transita, periodo di clock dopo periodo di clock, da uno stato all'altro, fino a tornare, dopo un certo periodo, che dipende dal seme inserito (cioè da come abbiamo settato i terminali di "tap" mediante i coefficienti binari C_i), allo stato iniziale caricato tramite le terminazioni C_i delle n porte AND.

Nel settore delle telecomunicazioni da terminale mobile ("wire-less communications") si utilizza spesso una particolare versione del LFSR, nota come PNSG, la quale costituisce la nostra scelta architetturale per la realizzazione del generatore di sequenze pseudo-casuali per trasmissioni CDMA IS-95. Qui di seguito riportiamo una rappresentazione astratta di un generico PNSG realizzato mediante l'architettura da noi scelta e descritta, mentre a pagina seguente abbiamo a destra la "schematic view", ossia lo schema circuitale dettagliato (da noi descritto in linguaggio VHDL), di uno PNSG a 15 stadi (15 registri flip-flop D) compattamente rappresentato dal polinomio caratteristico a coefficienti binari $G(x) = x^{15} + x^{13} + x^9 + x^8 + x^7 + x^5 + 1$, e a sinistra la "simbolic view" dello stesso PNSG, ossia la relativa black box, con le sole porte di ingresso e uscita, utilizzata nel test bench:




IR Initial Register
key (seme)
sel
init
clk
reset
STAGE1
STAGE2
STAGE3
STAGE4
STAGE5
STAGE6
STAGE7
STAGE8
STAGE9
STAGE10
STAGE11
STAGE12
STAGE13
STAGE14
STAGE15
D
Q
clk
reset
xor5
xor4
xor3
xor2
xor1
outxor1
inxor2
inxor1
PN_code
keystream
(codice pseudo-casuale)
LSB
MSB


IR
15
sel
reset
clk
PN_gen
15
PN_code

Funzionamento dell'architettura scelta per la realizzazione dello PNSG per trasmissioni CDMA IS-95

Di fondamentale importanza è assicurarsi che il segnale di reset, da noi preso attivo basso, comune ai 15 registri seriali realizzati mediante flip-flop D, abbia assunto stabilmente valore logico "1" nell'istante in cui, per la prima volta, il segnale di clock (comune anch'esso ai 15 registri in cascata) subisce una variazione di tipo positive edge, mentre il valore di "sel" ha già stabilmente assunto valore "0", ossia il loop di feedback è chiuso. Se nell'istante del primo fronte di salita del clock il reset fosse basso tutti i 15 flip-flop D aggiornerebbero le rispettive uscite Q, ciascuna di esse costituita da un singolo bit, con il valore logico "0", pertanto lo stato iniziale dello PNSG risulterebbe una stringa di 15 zeri e sarebbe impossibile per il dispositivo evolvere, nel tempo, in uno stato diverso, ossia uscire dallo stato iniziale caricato. Qualche istante prima del primo fronte di salita del clock è altresì necessario aver inserito il seme di 15 bit per mezzo degli appositi piedini e aver settato a "0" il bit di controllo "sel" (identificato come "init", nel codice VHDL) dei 15 multiplexer al fine di istruire gli stessi a fornire in uscita i bit costituenti la key appena inserita mediante i rispettivi canali di ingresso x_1. In tal modo al primo fronte di salita del clock il k-esimo flip-flop D ($\forall k: 1 \leq k \leq 15$) aggiorna la propria uscita Q_k con il valore del bit b_k imposto in ingresso dall'utente, dopodiché il bit "sel" deve essere reimpostato in modo da imporre ai multiplexer di fornire in uscita il valore del secondo ingresso x_0, ovvero l'uscita Q del registro precedente. Quindi dal secondo fronte di salita del clock in poi il generatore non risente più della word-line $b_1 \ldots b_{15}$ con la quale abbiamo inserito il seme (il disaccoppiamento fra la word-line di ingresso ed il circuito sottostante è garantito dallo stato $x_0 \rightarrow D$ in cui si trovano i 15 multiplexer) ed evolve, periodo di clock dopo periodo di clock, in funzione della logica implementata a livello circuitale, rappresentata compattamente dal polinomio caratteristico G(x).

A livello architetturale la struttura dello PNSG è molto simile a quella del LFSR: la differenza infatti consiste nelle modalità di inserimento del seme, effettuato appunto mediante i multiplexer e non attraverso le porte AND, e nel fatto che i bit C_i del LFSR sono realizzati rigidamente, cablati, mediante opportuni collegamenti circuitali statici, al fine di implementare la forma algebrica di G(x). Per esempio anziché imporre al bit C_{14} il valore logico "0" è sufficiente, per realizzare il coefficiente nullo che moltiplica il monomio x^{14} di G(x), evitare il collegamento fra Q_{14} e la porta XOR più vicina: infatti il risultato della AND_{14} sarebbe sempre 0, essendo $C_{14} = 0$, e di conseguenza l'uscita della XOR_{14} sarebbe 0 nel caso in cui l'uscita della XOR_{15} sia 0 ed 1 nel caso in cui l'uscita della XOR_{15} sia 1. Tale trasparenza della porta XOR_{14} nei confronti dell'output della porta XOR_{15}, ossia nei confronti del bit Q_{15} (poiché nella struttura equivalente del LFSR avremmo imposto $C_{15} = 1$), suggerisce la semplificazione circuitale che differenzia lo PNSG dal LFSR. Il risparmio in termini di numero di porte logiche AND e XOR è evidente.

Consideriamo adesso, in linea del tutto generale, uno PNSG nella sua forma "general purpose", vale a dire uno PNSG "LFSR – like", ossia completo di porte AND "tap" con le quali, mediante i bit C_i, è possibile settare il polinomio caratteristico G(x) che si desidera implementare: come già detto se $C_i = 1$ la porta AND – tap i-esima si comporta, di fatto, come un cortocircuito proveniente dal terminale Q_i, mentre se $C_i = 0$ la porta AND – tap i-esima si comporta come un circuito aperto (il

terminale Q_i non comunica con il piano circuitale delle porte XOR). Lo PNSR, formato da n registri, espande un seme lungo n bit, ossia una key di n bit da noi inserita, in una keystream di periodo massimo L = $(2^n - 1)$n bit, in $2^n - 1$ cicli di clock (n = numero dei registri seriali, cioè degli stadi), soltanto se sono verificate le seguenti due condizioni necessarie e sufficienti:

- il polinomio caratteristico G(x) implementato dallo PNSG deve essere "primo", ovvero non fattorizzabile
- il seme inserito $b_1 \ldots b_n$ deve corrispondere alla stringa 0…01

Ad esempio se lo PNSG costa di n = 15 stadi potremmo eseguire il seguente esperimento: inserire il seme 000000000000001 e permutare, di volta in volta, il valore 0/1 dei coefficienti binari di tap C_i, ovvero modificare la topologia circuitale equivalente del dispositivo, ossia modificare di volta in volta il polinomio caratteristico G(x) in modo tale che questo risulti primo, così da soddisfare le due condizioni di cui sopra. Ci accorgeremmo che per ciascuno dei 1800 G(x) primi implementabili mediante l'opportuno settaggio dei tap (settaggio opportuno dei bit C_i, 1800 combinazioni dei 15 bit C_i, ciascuna delle quali determina un particolare G(x) non fattorizzabile) avremmo, in uscita, una sequenza pseudo-casuale lunga L = $(2^n - 1)$n = $(2^{15} - 1)15$ = 491505 bit, ossia L = $2^{15} - 1$ = 32767 colpi di clock. Pertanto uno PNSG di 15 stadi ha la possibilità di generare, a partire dall'opportuno seme di cui sopra, ben 1800 keystream, 1800 chiavi, ovvero 1800 codici C (tutti diversi fra loro) di lunghezza pari al periodo L = 32767 T_{clock} = 491505 bit. Per calcolare il numero di codici C "massimamente lunghi" (cioè di lunghezza pari al periodo L = $2^n - 1$), dato uno PNSG ad n stadi, è sufficiente applicare la seguente formula:

$$\text{numero dei codici C} = \frac{1}{n}\prod_j \left\{ {P_j}^{(\alpha_j - 1)}(P_j - 1) \right\} = \text{numero dei G(x) primi}$$

dove i P_j costituiscono i numeri primi (3, 5, 7, 11 ecc…) che scompongono il numero che rappresenta, in periodi di clock, il periodo L = $2^n - 1$, mentre gli α_i sono gli esponenti di tali fattori primi. Per esempio se considerassimo uno PNSG di n = 6 stadi, avremmo periodo L = $2^n - 1 = 2^6 - 1 = 63$ T_{clock}, pertanto settando tale dispositivo con gli opportuni coefficienti binari di tap avremmo che, in risposta al seme 000001, la sequenza pseudo-casuale di uscita sarebbe costituita da 378 bit.

$$63 = 3\text{x}3\text{x}7 = 3^2\text{x}7 = {P_1}^{\alpha_1} x\ {P_2}^{\alpha_2} \rightarrow P_1 = 3,\ \alpha_1 = 2,\ P_2 = 7,\ \alpha_2 = 1$$

Quindi:

$$\text{numero dei codici C} = \frac{1}{6}{P_1}^{(\alpha_1 - 1)}(P_1 - 1)\ {P_2}^{(\alpha_2 - 1)}(P_2 - 1) = \frac{1}{6}3^{(2-1)}(3 - 1)\ 7^{(1-1)}(7 - 1) = 6$$

Pertanto il numero di G(x) primi, settabili mediante opportune combinazioni dei bit C_i di tap, sono 6 (ci sono 6 combinazioni dei valori 0/1 assegnabili alla word-line $C_1 \ldots C_6$ che realizzano uno G(x) primo). In altri termini lo PNSG a 6 stadi può fornire, in risposta al seme 000001, 6 sequenze pseudo-casuali C(t) massimamente lunghe, ovvero 6 codici lunghi 378 bit. Qui di seguito riportiamo una tabella riassuntiva dei risultati finora visti:

numero n degli stadi	lunghezza in colpi di clock del periodo L	numero dei codici C massimamente lunghi	alcune configurazioni di taps per la realizzazione del feedback
2	3	1	[2, 1]
3	7	2	[3, 2] [3, 1]
4	15	2	[4, 3] [4, 1]
5	31	6	[5, 3] [5, 2]
6	63	6	[6, 5] [6, 1]
7	127	18	[7, 6] [7, 3] [7, 1]
8	255	16	[8, 6, 5, 4] [8, 6, 5, 3]
9	511	48	[9, 5] [9, 6, 4, 3]
10	1 023	60	[10, 7] [10, 3]
11	2 047	176	[11, 9] [11, 8, 5, 2]
12	4 095	144	[12, 6, 4, 1]
13	8 191	630	[13, 4, 3, 1]
14	16 383	756	[14, 5, 3, 1]
15	32 767	1 800	[15, 14] [15, 4]
16	65 535	2 048	[16, 15, 13, 4]
17	131 071	7 710	[17, 14] [17, 3]
18	262 143	7 776	[18, 11[[18, 7]
19	524 287	27 594	[19, 6, 2, 1]
20	1 048 575	24 000	[20, 17] [20, 3]
21	2 097 151	84 672	[21, 19] [21, 2]
22	4 194 303	120 032	[22, 21] [22, 1]
23	8 388 607	356 960	[23, 18] [23, 5]
24	16 777 215	276 480	[24, 23, 22, 17]
25	33 554 431	1 296 000	[25, 22] [25, 3]

Il generatore da noi descritto in VHDL implementa un polinomio caratteristico G(x) non scomponibile, ed inserendo nel test bench il particolare seme sopra riportato abbiamo constatato che la periodicità della sequenza pseudo-casuale in uscita è proprio quella massima prevista, ovvero $L = 2^n - 1 = 2^{15} - 1 = 32767\ T_{clock}$.

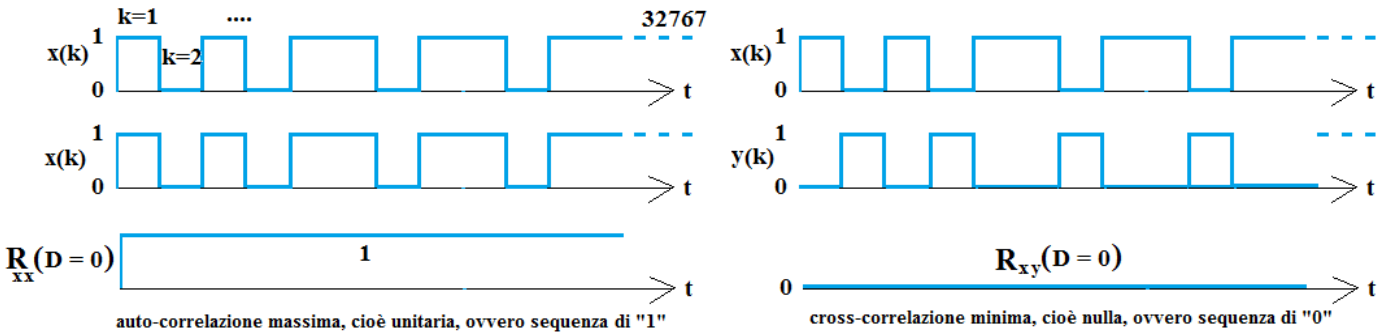
Delle 1800 sequenze pseudo-casuali, lunghe 32767 T_{clock}, che il nostro PNSG a 15 stadi può erogare, in risposta al seme 000000000000001, una per ciascuno dei 1800 G(x) primi settabili, 512 hanno una particolare proprietà: sono ortogonali nel tempo. Ciò significa che la funzione di correlazione R_{xy} fra due qualsiasi di quelle 512 sequenze, per esempio fra il codice x(k) ed il codice y(k) (k = indice di bit = (1, 2, 3, …, L – 1 = 491505)), è nulla, se x è diverso da y, cioè se la correlazione è fra due codici diversi (si parla di "cross-correlazione" minima, cioè nulla), mentre la funzione di correlazione fra un codice qualunque x(k), uno qualunque di quei 512, con se stesso, è unitaria (si parla di "auto-correlazione" massima, cioè unitaria). Quei 512 codici formano un set di sequenze perfettamente orto-normali e si usano largamente, come vedremo, nelle comunicazioni CDMA.

Formalmente la funzione di correlazione (cross se $x \neq y$, auto se x = y) $R_{xy}(D)$, in funzione cioè del ritardo relativo D (= delay) fra il codice binario x(k) e quello y(k) è definita come:

$$R_{xy}(D) = \sum_{k=1}^{L} x(k)\, y(k+D) = \text{XNOR}_{bit\_wise}(x,y) = \langle x(k) \mid y(k) \rangle$$

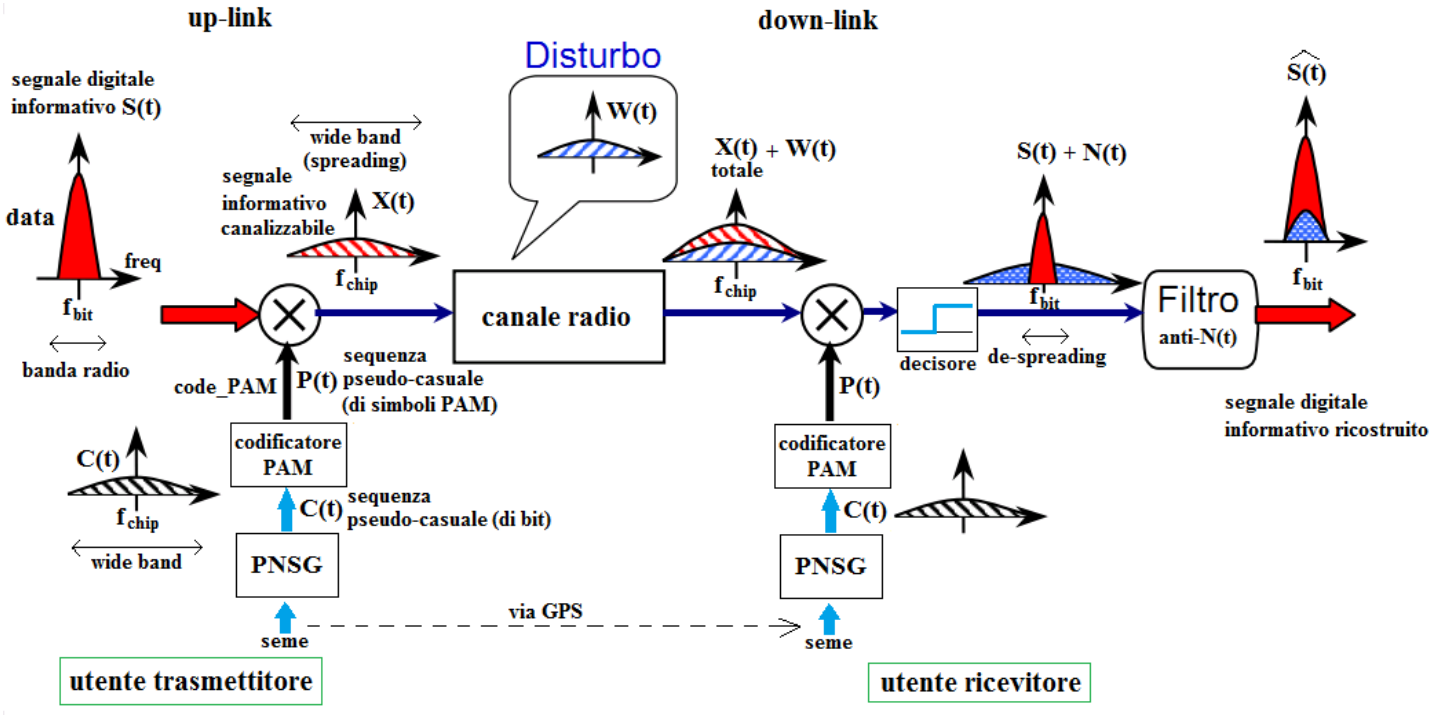
La funzione di correlazione fra due stringhe di bit, fra due codici pseudo-casuali C ortogonali, che può essere associata ad un integrale di overlap (di sovrapposizione), ossia ad un prodotto scalare fra segnali rappresentabili come funzioni del tempo appartenenti allo spazio vettoriale di Hilbert,

non è altro che una sequenza di bit data dalla somma modulo-2, eseguita bit a bit ("bit-wise"), fra la sequenza x(k) e quella y(k), con k = 1, 2, 3, …, L = $2^n - 1 = 2^{15} - 1 = 491505$ bit. La somma modulo-2 corrisponde, nel caso appunto di cifre binarie, all'operazione di OR esclusiva negata (NXOR = $\overline{XOR}$) bit a bit, che ritorna 1 se i due bit k-esimi, uno del codice x, l'altro di quello y, sono uguali, 0 se sono diversi, come mostrato nella seguente illustrazione:

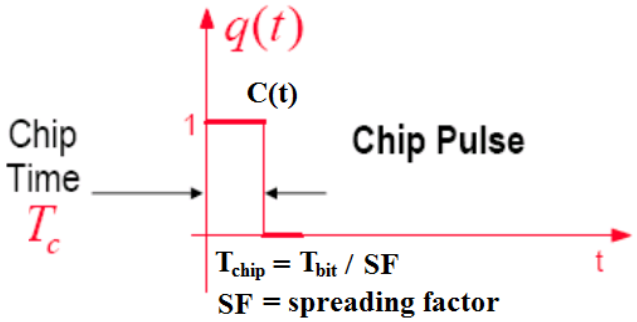


Applicazione dello PNSG progettato alle telecomunicazioni cellulari CDMA: concetti di base

Il generatore di sequenze pseudo-casuali da noi progettato e descritto in VHDL (15 stadi, polinomio caratteristico G(x) = $x^{15} + x^{13} + x^9 + x^8 + x^7 + x^5 + 1$) trova una delle sue applicazioni più importanti nella telefonia cellulare di terza generazione, ovvero nelle comunicazioni da terminale mobile che usufruiscono dell'innovativa tecnica di accesso radio nota come UMTS (Universal Mobile Telecommunication System). Mentre la tecnica GSM è basata su un tipo di accesso al canale radio che si avvale di una combinazione fra multiplazione a divisione di tempo ("Time Division Multiple Access", TDMA) e multiplazione a divisione di frequenza ("Frequency Division Multiple Access", FDMA), nell'UMTS viene introdotto l'accesso a divisione di codice ("Wideband Code Division Multiple Access", W-CDMA). Tale tecnica permette agli utenti di trasmettere sulla stessa frequenza e nello stesso tempo, pertanto è necessario prevedere un meccanismo numerico, a livello di codice, per realizzare un'efficiente separazione dei vari utenti. Facendo riferimento ai seguenti disegni illustriamo il concetto che sta alla base della tecnica W-CDMA:

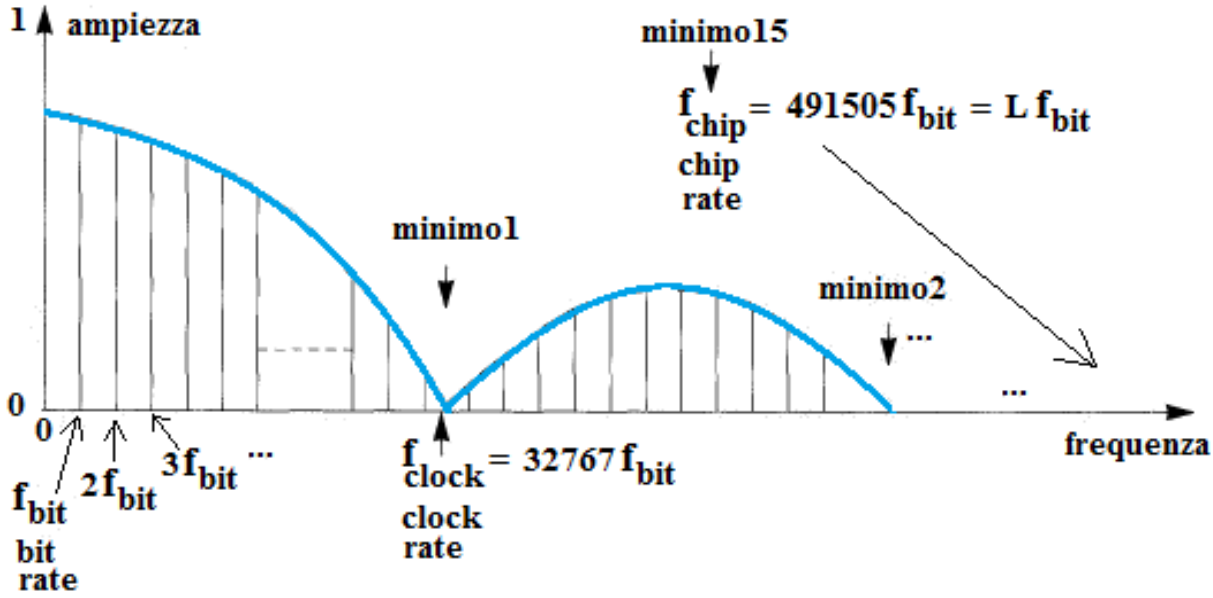


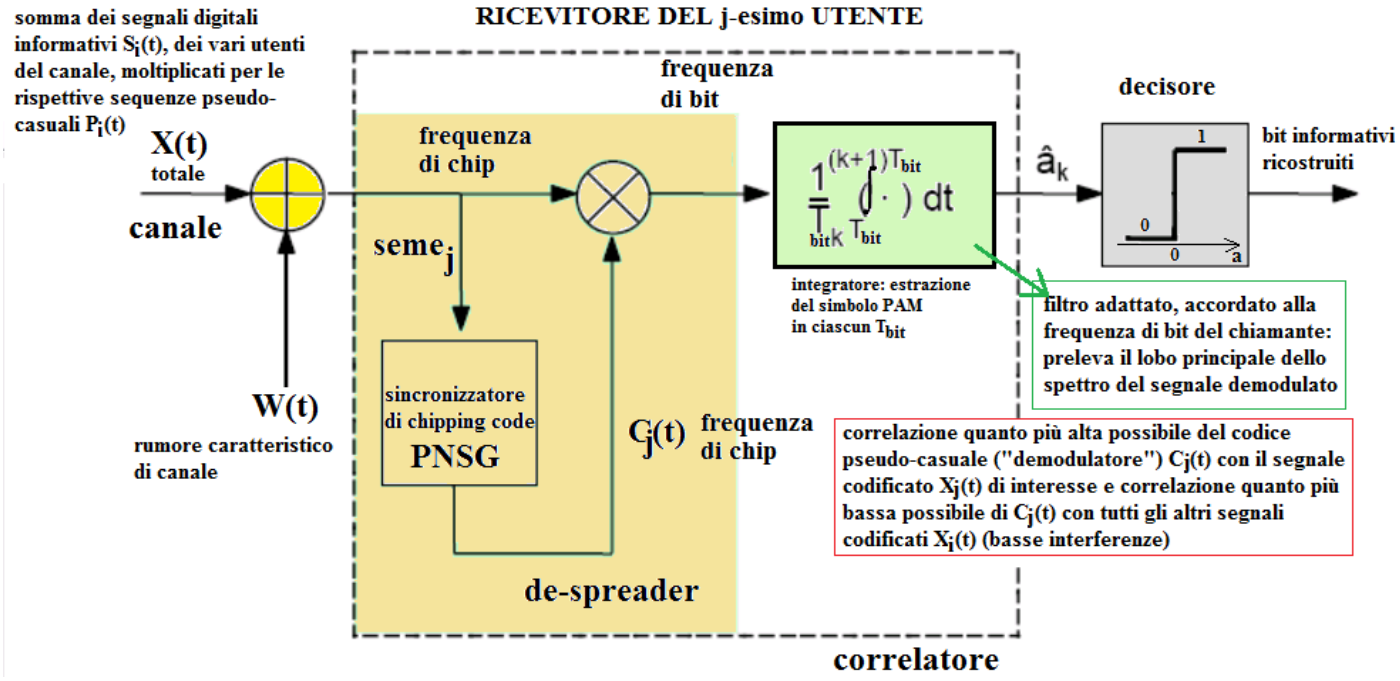
PSEUDO-NOISE Sequence

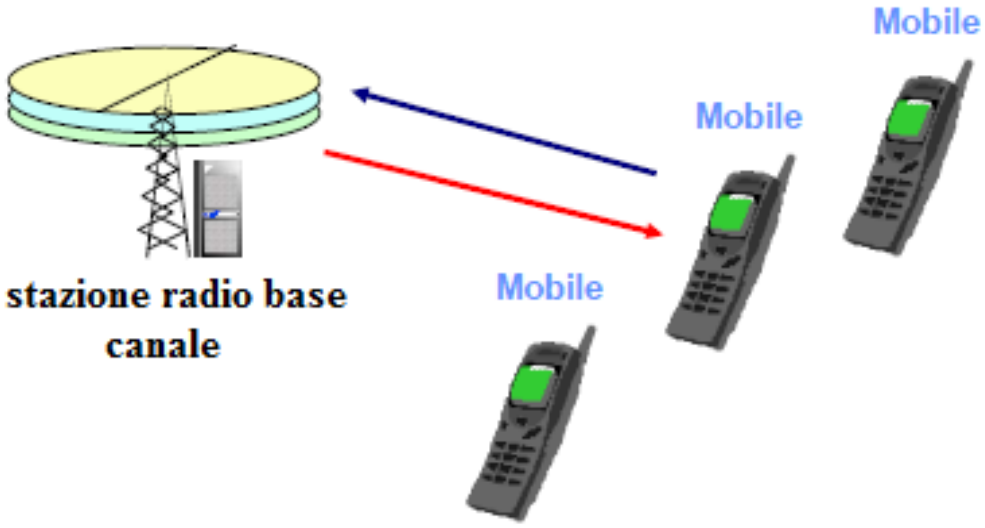


$$c(t) = \sum_{m=-\infty}^{\infty} c_m q(t - mT_c)$$

Spettro del segnale pseudo-casuale generato dal nostro PN_gen a 15 stadi







Supponiamo che vari utenti, che rappresentano le diverse sorgenti di segnale, i cui cellulari aggancino una medesima cella radio (la stessa stazione radio base, ovvero lo stesso canale), cerchino di trasmettere simultaneamente le rispettive telefonate, ossia di accedere nello stesso tempo al canale radio comune (tale situazione è nota come "up-link"). Evidenziamo subito che tale situazione è del tutto teorica, dal momento che ciascuna telefonata è temporalmente scorrelata e indipendente l'una dall'altra, ossia ciascuna comunicazione inizia, in generale, ad un istante diverso dalle altre, tuttavia tale simultaneità ci è utile per esporre i concetti di base, pertanto la assumiano, almeno per il momento, come ipotesi di lavoro. Ciascun terminale mobile trasmette in banda radio un certo segnale digitale, ossia una stringa di bit rappresentativa del contenuto informativo da trasmettere al rispettivo ricevitore. Tale segnale digitale S(t) occupa una banda radio approssimativamente centrata sulla frequenza f_{bit} (= "bit rate") = $1/T_{bit}$, dove T_{bit} rappresenta la durata del singolo bit informativo trasmesso. La stringa di bit "data" di ciascuna sorgente di segnale viene battuta su un mixer con un segnale "code_pam" risultante dalla codifica 2-PAM della sequenza di bit pseudo-casuale (keystream) generata da uno PNSG, a partire da un particolare seme (livello logico "0" del codice pseudo-casuale → simbolo 2-PAM "+1", livello logico "1" del codice pseudo-casuale → simbolo 2-PAM "-1"). Ad ogni utente che accede al canale radio comune viene assegnato un seme diverso, una key diversa, ossia un codice iniziale diverso di 15 bit; questo seme viene tradotto, espanso, in una keystream, cioè in un codice pseudo-casuale molto più lungo dallo PNSG che serve ciascun utente. Pertanto a ciascun utente viene associata una sequenza pseudo-casuale diversa, derivante da una key diversa, e tale sequenza costituisce un segnale digitale C(t) che, una volta codificato in un segnale 2-PAM P(t), va a convolvere nel tempo con il segnale informativo costituito dal flusso di bit S(t): il risultato di tale convoluzione è il segnale X(t). Il segnale C(t) è costituito da una sequenza temporale randomica (pseudo-randomica) di bit di canalizzazione, detti "chip", e da una frequenza centrale di banda f_{chip} (= "chip rate") = $1/T_{chip}$ molto maggiore della f_{bit}, ossia ciascun chip di C(t) ha una durata T_{chip} molto inferiore a quella T_{bit} del singolo bit informativo da trasmettere. Le osservazioni fatte per C(t) valgono anche per la versione 2-PAM di C(t), ovvero P(t). Pertanto un singolo bit di informazione, di durata T_{bit}, trasmesso da un particolare utente, è modulato, codificato, da una lunga sequenza di chip 2-PAM (numero di chip >> 15), è tale sequenza è assegnata univocamente a quel particolare utente per mezzo del funzionamento dello PNSG, che appunto espande il seme di 15 bit, univocamente assegnato all'utente, in una sequenza di canalizzazione C(t), poi tradotta in P(t). Supponiamo che ciascun singolo bit informativo, trasmesso da un particolare utente, sia modulato dall'intera sequenza di chip P(t) di periodo M ($M \leq L$ = periodo massimo = 491505 chip), ovvero che $T_{bit} = M\ T_{chip}$ (SF = "Spreading Factor" = M), e quindi che il chip rate sia M volte il bit rate, ossia che la

portante f_{chip} sia M volte superiore a f_{bit}. Inoltre abbiamo che la banda occupata dal segnale X(t) è molto più larga (indicativamente M volte più larga) della banda occupata dal segnale S(t), e ciò, come vedremo, rende il segnale canalizzato X(t) molto robusto nei confronti del rumore di canale. P(t) è detta "sequenza di spreading o di canalizzazione", proprio per la sua proprietà di allargare lo spettro del segnale digitale S(t) con cui P(t) fa battimento. Assumendo per adesso, in modo del tutto ideale, che i vari utenti della cella radio (del canale) telefonino a partire dallo stesso istante iniziale, ovvero che i vari codici pseudo-casuali $P_i(t)$ siano perfettamente sincroni, e che magari ciascuno di loro sia generato da uno PNSG a 15 stadi descritto da un particolare polinomio caratteristico G(x) primo, si può affermare che i segnali $P_i(t)$, detti anche "spreading vectors" o "chipping codes", costituiscono una base ortonormale di codici massimamente lunghi (M = L = 491505 bit), ovvero che la funzione di cross-correlazione è nulla e che quella di auto-correlazione è unitaria. In altre parole i codici pseudo-casuali $P_i(t)$ sono ortogonali fra loro nel tempo, mentre in frequenza sono di fatto sovrapposti, dal momento che il chip rate dei vari utenti è lo stesso; va detto altresì che il bit rate può variare da utente a utente in base alla larghezza della banda base del segnale sorgente di ciascun utente (e quindi in base alla frequenza di campionamento a cui lavora il convertitore analogico/digitale di ciascun terminale mobile), pertanto ne consegue che lo spreading factor SF può variare da utente a utente. In formule si ha:

$$\langle \mathrm{P_i(t)} \,|\, \mathrm{P_j(t)} \rangle = \frac{1}{\mathrm{MT_{chip}}} \int_{t\,=\,k\mathrm{T_{bit}}}^{t\,=\,k\mathrm{T_{bit}}+\,\mathrm{MT_{chip}}} \mathrm{P_i(t)\ P_j(t)\ dt} = \delta_{Kronecker} = \begin{cases} 1\ se\ i = j \\ 0\ se\ i \neq j \end{cases}$$

per ogni k-esimo bit del flusso informativo totale (= somma dei flussi informativi trasmessi dai singoli utenti) che transita nel canale radio comune.
Nel canale di fatto transitano vari segnali $X_i(t)$, ciascuno dei quali ha subito un allargamento spettrale grazie al codice pseudo-casuale $P_i(t)$; anche i segnali $X_i(t)$ sono ortogonali fra loro nel tempo. Tale ortogonalità costituisce la risorsa che consente la separazione dei vari utenti a livello di stadio ricevitore, così che ciascun ricevitore possa demodulare e quindi riportare in chiaro solo il segnale informativo di interesse, eludendo efficacemente l'interferenza con tutti gli altri segnali che hanno acceduto simultaneamente al canale comune.
Supponiamo che un particolare utente mobile k-esimo desideri selezionare il segnale $X_k(t)$ e demodularlo, così da estrarre il segnale informativo di interesse $S_k(t)$. Supponiamo, come già detto, che non vi siano problemi di asincronismo fra gli spreading vectors che modulano, codificandoli, i vari segnali $X_i(t)$ transitanti, pertanto gli stessi $X_i(t)$ risultano ortogonali fra loro nel tempo. L'antenna del ricevitore mobile k-esimo riceve e trasduce in segnali elettrici di fatto tutto il segnale elettromagnetico $X_{totale}(t)$ transitante nel canale radio comune, dato da:

$$\mathrm{X_{totale}(t)} = \sum_{i=1}^{Numero\ utenti\ canale} X_i(t) \ +\ \mathrm{W(t)}$$

dove W(t) rappresenta il rumore gaussiano bianco caratteristico del canale radio. Attraverso un canale parallelo ritenuto sicuro ed affidabile, come ad esempio il sistema GPS (vedremo in seguito che è proprio questo a sovraintendere la sincronizzazione, cioè l'ortogonalizzazione, dei codici di canalizzazione delle varie stazioni radio base sparse sul territorio), il trasmettitore k-esimo invia il particolare seme Key_k di 15 bit utilizzato per produrre, in fase di up-link, mediante lo PNSG, il codice pseudo-casuale $C_k(t)$, al ricevitore k-esimo con il quale l'utente k-esimo del canale desidera comunicare. In altre parole si ha il caricamento, di fatto simultaneo, di uno stato iniziale comune (il seme Key_k) ai due stadi di end del sistema (trasmettitore k-esimo o "front-end" e ricevitore k-esimo o "back-end"). Via GPS la chiave Key_k arriva allo PNSG dello stadio "despreader" del ricevitore k-esimo, lo PNSG, così caricato, produce la stessa sequenza pseudo-random di bit $C_k(t)$

utilizzata in fase di up-link. $C_k(t)$ viene codificata dal codificatore 2-PAM nella sequenza di despreading $P_k(t)$, e questa viene battuta sul mixer del despreader con il segnale totale di canale $X_{totale}(t)$. Il segnale uscente dal mixer è:

$$\widehat{S_k}(t) = \langle X_{tot}(t) \,|P_k(t)\rangle = S_k(t) + N(t)$$

dove $S_k(t)$ è il segnale digitale demodulato, cioè decodificato (questa fase è chiamata "down-link"), ossia la sequenza temporale di bit in chiaro inviata al canale dal trasmettitore k-esimo, avente portante f_{bit}, mentre N(t) rappresenta un rumore dato dalla somma di tutte le sorgenti di rumore possibili riscontrabili in uscita dallo stadio di despreading, ovvero il residuo del rumore caratteristico di canale, il rumore derivante da inevitabili interferenze fra gli utenti del canale stesso (da noi trascurate fino ad ora), il rumore di antenna, di amplificazione a radiofrequenza, di mixer ecc… Uno stadio di filtraggio consente di attenuare il rumore N(t) ed ottenere un segnale digitale $\widehat{S_k}(t)$ molto simile a quello trasmesso $S_k(t)$.
Qui di seguito proponiamo un esempio ipersemplificato dei concetti espressi finora. Fra i vari utenti di canale serviti da una certa cella radio supponiamo che ce ne siano due, l'utente i-esimo e l'utente j-esimo, i quali trasmettono rispettivamente i segnali digitali $S_i(t)$ e $S_j(t)$, qui di seguito riportati:

$$S_i(t) = (1,0,1,1) \qquad S_j(t) = (0,0,1,1)$$

Allo PNSG dello i-esimo trasmettitore viene assegnato il seme Key_i, mentre allo PNSG dello j-esimo trasmettitore viene assegnato il seme Key_j, pertanto il generatore dello i-esimo utente produce il chipping code 2-PAM codificato $P_i(t) = (1, –1)$, mentre il generatore dello j-esimo utente produce il chipping code 2-PAM codificato $P_j(t) = (1, 1)$. $P_i(t)$ è ortogonale nel tempo a $P_j(t)$. I segnali informativi digitali $S_i(t)$ e $S_j(t)$ vengono mixati con le rispettive sequenze pseudo-casuali di spreading $P_i(t)$ e $P_j(t)$, tenendo conto che i due utenti desiderano optare per uno spreading factor SF pari a 2, ovvero codificare (modulare) ciascun bit delle rispettive sequenze digitali informative $S_i(t)$ e $S_j(t)$ con soli 2 chip ($T_{bit} = M\, T_{chip} = 2\, T_{chip}$, il chip rate è il doppio del bit rate), ovvero con le rispettive sequenze randomiche intere $P_i(t)$ e $P_j(t)$ di periodo M = 2. Assumiamo che qualunque sia il chipping code V (ad esempio $P_i(t)$, $P_j(t)$, … , $P_n(t)$, …), ovvero qualunque sia l'utente del canale radio comune, se il dato da trasmettere è uno "0" logico la stringa di chip da inviare al canale corrisponderà al codice –V, mentre se il dato da trasmettere è un "1" logico la stringa di chip da inviare corrisponderà al codice V. In altre parole il battimento nel mixer fra "0" e V fornisce –V, mentre il battimento fra "1" e V fornisce V.

In fase di up-link (spreading dei segnali digitali informativi S(t)) si ha:

$$S_i(t) \otimes P_i(t) = (1,0,1,1) \otimes (1, –1) = [(1, –1), (–1, 1), (1, –1), (1, –1)] = X_i(t) \text{ (frequenza = } f_{chip})$$

$$S_j(t) \otimes P_j(t) = (0,0,1,1) \otimes (1, 1) = [(–1, –1), (–1, –1), (1, 1), (1, 1)] = X_j(t) \text{ (frequenza = } f_{chip})$$

Nel canale i segnali modulati (codificati) si sommano:

$$X_{tot}(t) = X_i(t) + X_j(t) = [(0, –2), (–2, 0), (2, 0), (2, 0)] \text{ (frequenza = } f_{chip})$$

In fase di down-link (despreading, demodulazione, decodifica), nel caso si sia interessati a ricevere il segnale $X_i(t)$, avremo:

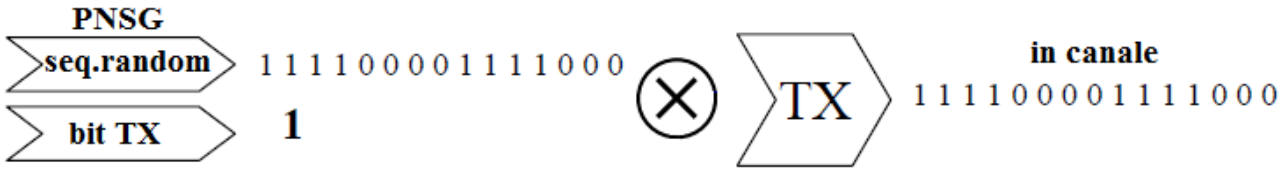
$$X_{tot}(t) \otimes P_i(t) = [(0, -2), (-2, 0), (2, 0), (2, 0)] \otimes (1, -1) \rightarrow \text{integratore} \rightarrow (2, -2, 2, 2)$$
(frequenza = f_{bit} nuovamente)

Non rimane che introdurre l'ultima stringa numerica, fornita dall'integratore, in ingresso ad un decisore a soglia che fornisca in uscita il valore logico "1" per qualunque valore positivo del segnale di input (= simboli PAM) ed il valore logico "0" per qualunque valore negativo. Si ha dunque:

$$(2, -2, 2, 2) \rightarrow \text{decisore} \rightarrow (1,0,1,1) = S_i(t)$$

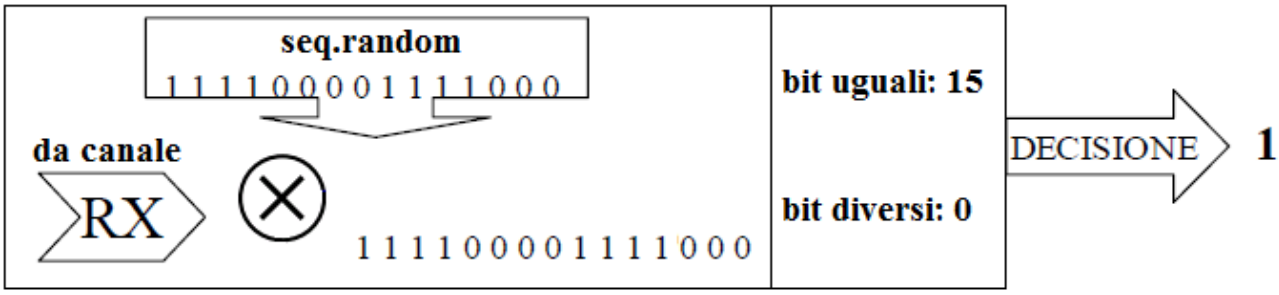
Esponiamo adesso molto brevemente, facendo ricorso ad un altro semplice esempio, come la tecnica CDMA sia adatta nel caso di canale radio molto rumoroso. Non perdiamo in generalità nel supporre che il chipping code P(t), che determina l'azione di spreading spectrum del segnale digitale informativo S(t) da trasmettere, non sia modulato da simboli 2-PAM (+1, –1) come finora supposto, bensì da semplici bit, ciascuno di durata T_{chip} (in altre parole P(t) = C(t)). Supponiamo altresì che lo spreading factor SF sia pari a 15 (T_{bit} = 15 T_{chip}), pertanto per ciascun utente agganciato alla stazione radio base comune la stringa di bit da trasmettere viene fatta convolvere (battere) con una particolare sequenza pseudo-casuale assegnata, per mezzo dell'algoritmo implementato dallo PNSG, all'utente stesso. Potrebbe essere utilizzato, in questo semplice esempio, uno PNSG formato da n = 4 stadi, e a seconda del particolare settaggio dei 4 bit di tap, impostato dall'utente, avremo due possibili G(x) primi e due possibili codici massimamente lunghi, ovvero lunghi ($2^4 - 1$)4 = 60 bit (= periodo L = $2^4 - 1$ = 15 periodi di clock), ortogonali fra loro. Ciascun bit di informazione potrebbe essere codificato, in linea di principio, da una sequenza pari ad un quarto di quella massimamente lunga, ovvero da 15 chip: il valore logico informativo "1" è codificato con la sequenza pseudo-random assegnata all'utente, mentre il valore logico "0" è codificato con la sequenza pseudo-random complementare. Supponiamo di analizzare il sistema durante il lasso di tempo T_{bit} nel quale un particolare utente (in questo caso uno dei due utenti di canale separabili, i cui codici cioè sono ortogonali, scorrelati fra loro) trasmette un bit di valore "1":

A livello di up-link si ha:

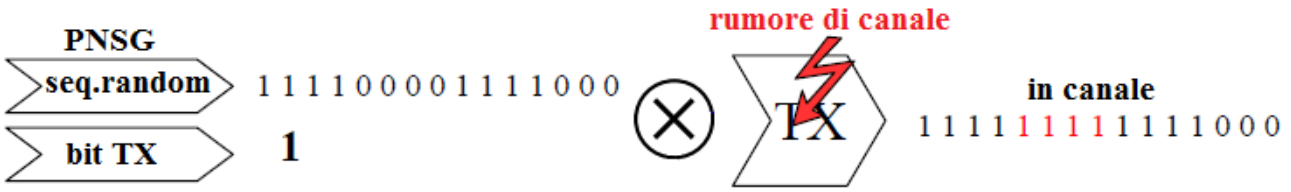


A livello di down-link il ricevitore CDMA, il cui funzionamento deve essere opportunamente regolato (scandito) da un segnale di clock di frequenza pari a quella di chipping (che al contrario di quella di bit è la stessa per tutti i segnali transitanti nel canale), deve effettuare una comparazione fra ciascun chip (0/1) della sequenza pseudo-casuale C(t) costituente il codice di despreading ed il corrispondente chip codificante l'informazione digitale che si desidera correlare con C(t), ovvero selezionare per la demodulazione. È necessario un registro R_{RX} in grado di immagazzinare il risultato logico del confronto (ad esempio "1" se i due chip sono uguali, "0" se sono diversi, e tale funzione è implementabile con una porta XNOR). Alla fine del confronto fra la sequenza di chip proveniente dal canale e la sequenza di chip generata localmente dallo PNSG che appartiene al

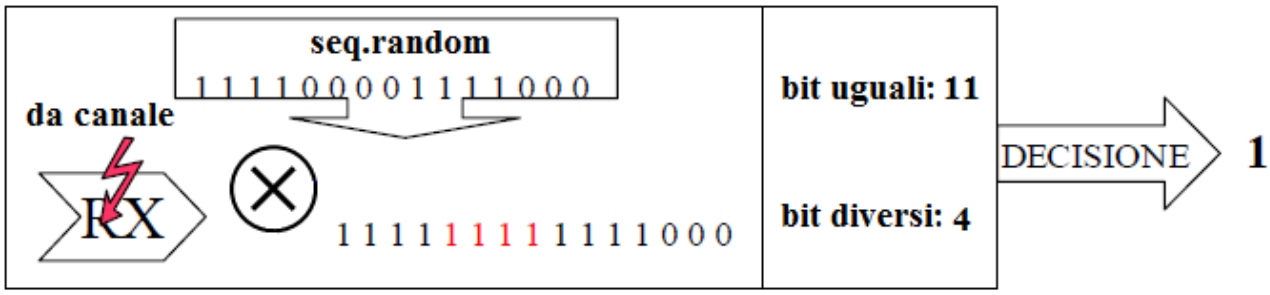
circuito di despreading del ricevitore, il registro R_{RX} avrà memorizzato una stringa di 15 “1” oppure di 15 zeri, a seconda che l’informazione trasmessa fosse il bit 1 oppure 0.



Supponiamo adesso che il canale introduca del rumore e che questo determini un’alterazione del contenuto informativo del flusso di chip trasmesso, come mostrato in figura:



A livello di ricevitore CDMA è possibile utilizzare il registro R_{RX} sopra citato per reiettare efficacemente la BER (Bit Error Rate) che affligge la corretta ricostruzione del segnale digitale informativo S(t):



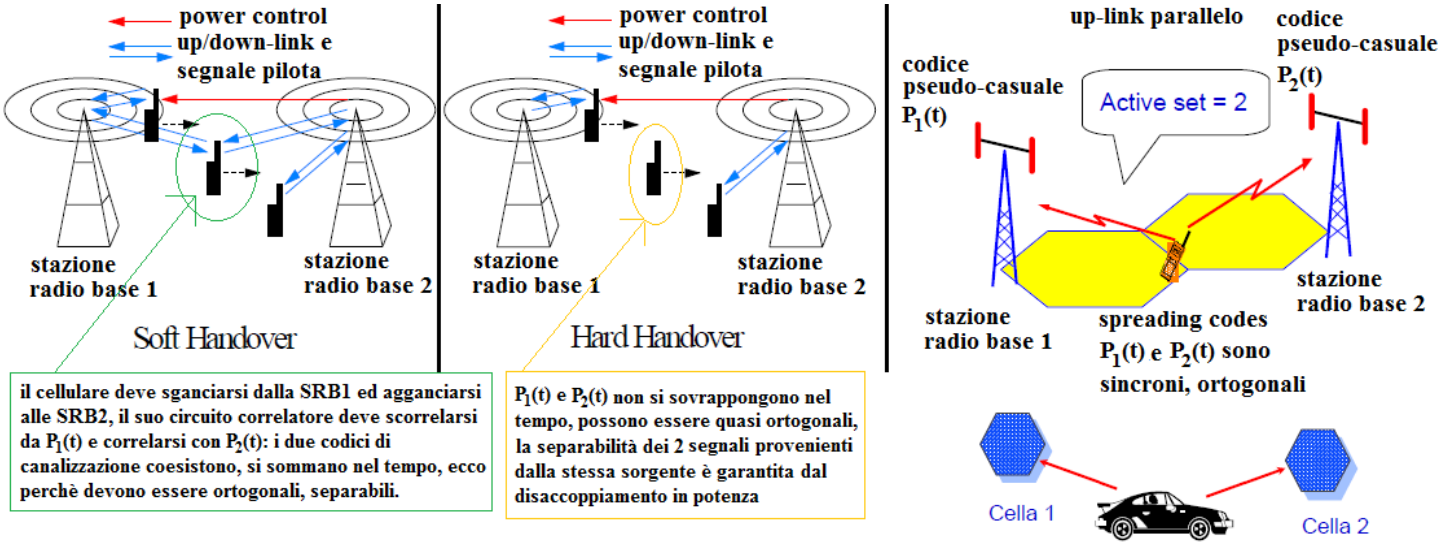
Nel nostro esempio il registro R_{RX} contiene, alla fine del confronto fra la sequenza di chip (rumorosa) proveniente dal canale e la sequenza di chip di despreading (chipping code), una stringa di 11 “1” ed una di 4 “0”, ovvero l’operazione seriale (scandita dal clock) di comparazione fra chip e chip ha riscontrato un numero di eventi “chip uguali” decisamente maggiore del numero di eventi “chip diversi” (11 vs 4), pertanto il decisore fornisce, come stima (ricostruzione) del bit trasmesso dalla sorgente mobile di segnale, il valore logico “1”. Se all’interno della slot temporale coincidente con la durata T_{bit} di un bit informativo, ossia coincidente con 15 intervalli di segnalazione T_{chip}, il numero di eventi “chip uguali” è maggiore del numero di eventi “chip diversi”, il decisore fornisce in ingresso all’integratore il valore logico “1”, viceversa “0”. Il massimo rumore tollerato dal sistema (margine di affidabilità), prima che il decisore produca una stima errata del bit trasmesso, è pertanto proporzionale allo spreading factor SF utilizzato dall’utente, ovvero al periodo M (in questo caso semplificato SF = M = 15) della sequenza di canalizzazione C(t) assegnata all’utente; il massimo rumore di canale tollerato dal sistema è quello che determina un’alterazione di (M + 1)/2 (= 8 in questo caso) chip per singolo bit trasmesso. Ne discende che per

aumentare l'affidabilità (la robustezza verso il rumore) del sistema CDMA in canali molto rumorosi può risultare conveniente adottare protocolli di trasmissione che prevedono keystream di spreading molto lunghe (grandi periodi M) e spreading factors SF, per i vari utenti, pari alla lunghezza di periodo dei codici di chipping. Tale esigenza, tuttavia, contrasta con la necessità di mantenere un throughput accettabile: maggiore è la robustezza, minore è il throughput, dunque minore è la velocità di ricezione dei dati.

Precisazioni sul reale funzionamento delle comunicazioni CDMA: codici di spreading e codici di scrambling

Nella realtà dei fatti alle comunicazioni UMTS, secondo la tecnica W-CDMA, prendono parte due famiglie di codici pseudo-casuali, generabili da uno PNSG come quello da noi descritto in VHDL: i codici di spreading e quelli di scrambling.
Supponiamo di avere un certo numero di cellulari tutti spazialmente vicini, ciascuno dei quali è servito da un certo numero di stazioni radio base, ovvero ciascun terminale mobile, ciascun utente trasmettitore, possiede un "active set" > 1; per "active set" si intende il numero delle stazioni radio base dalle quali uno stesso cellulare riceve un segnale pilota di potenza maggiore di una certa soglia. Supponiamo che uno di questi trasmettitori mobili (sorgenti di segnale digitale $S_i(t)$) inizi a trasmettere la sequenza di bit di informazione avvalendosi del supporto (supporto = ripetizione del segnale) delle varie stazioni radio base facenti parte, in quel momento, del proprio active set. Ciascuna di queste stazioni radio base associa al segnale $S_i(t)$, cioè all'utente in questione, un particolare codice pseudo-casuale $P_i(t)$. I vari codici $P_i(t)$ (i = 1, 2, … , AS, dove AS = active set del singolo utente in questione) sono ortogonali fra loro poiché il segnale digitale $S_i(t)$ che queste sequenze pseudo-random codificano è lo stesso, trasmesso dallo stesso cellulare, pertanto i vari $P_i(t)$ (uno per ciascun canale, ovvero uno per ciascuna stazione radio base) sono tutti perfettamente sincronizzabili a livello di sistema GPS e quindi l'ortogonalità è garantita. Il sistema GPS, conoscendo la posizione di queste stazione radio base sul territorio e di conseguenza conoscendo i ritardi di propagazione relativi fra i segnali $S_i(t)$ provenienti da una stessa sorgente mobile (le "copie" $S_i(t)$ della comunicazione) che aggancia in parallelo quelle stazioni radio, riesce a settare i vari PNSG, e quindi i vari codici di canalizzazione massimamente lunghi, in modo da renderli perfettamente ortogonali ($R_{xy}(D) = 0\ \forall x,y$), ergo separabili fra loro. Ciò consente di separare i segnali provenienti dalla singola sorgente mobile in questione.
L'importanza di tale separazione si palesa in fase di down-link, ovvero durante le trasmissioni dalle stazioni radio base ai ricevitori mobili. Supponiamo infatti che l'utente trasmettitore in questione stesse comunicando con un particolare utente ricevitore mobile. Quest'ultimo riceve, durante la comunicazione con il trasmettitore mobile di prima, il proprio segnale pilota da un certo numero Y di stazioni radio base, ossia aggancia, in quei momenti, Y celle diverse: ciò significa che la sua antenna patch riceve Y segnali provenienti dal trasmettitore mobile, ovvero Y "copie" $X_i(t)$ (i = 1, 2, … , Y) della comunicazione che il trasmettitore sta inviando in parallelo ai vari canali (alle varie stazioni radio base), ciascuna spettralmente allargata dal rispettivo codice di spreading $P_i(t)$. I vari codici $P_i(t)$ sono ortogonali, pertanto ottimamente separabili. Nel lasso di tempo in cui il terminale mobile ricevitore si trova nella regione di territorio servita meglio, dal punto di vista della potenza trasmessa, dalla stazione radio base 1, il suo circuito correlatore tenterà di correlare ottimamente ($\langle X|P\rangle \rightarrow 1$) il codice $P_1(t)$ con il segnale $X_1(t)$ e di scorrelare altrettanto ottimamente ($\langle X|P\rangle \rightarrow 0$) $P_1(t)$ da tutti gli altri Y – 1 segnali codificati $X_i(t)$, ovvero da tutti gli altri Y – 1 codici di spreading $P_i(t)$. In altre parole soltanto $X_1(t)$ costituisce il segnale codificato utile, quello gradito, mentre tutti gli altri Y – 1 segnali codificati $X_i(t)$, provenienti dal medesimo cellulare trasmettitore, sono considerati come dei disturbi, dei rumori, delle interferenze da reiettare quanto più possibile: tale

reiezione è possibile in virtù del sincronismo dei vari segnali $X_i(t)$ (ovvero dei vari codici di spreading $P_i(t)$) provenienti dalla medesima sorgente (il cellulare chiamante). Ad un certo istante t_S il cellulare ricevente passa dalla regione territoriale di cui sopra ad una, contigua, servita meglio da un'altra stazione radio base, ad esempio dalla stazione radio base 2. Durante il lasso di tempo, successivo a t_S, nel quale il cellulare ricevente rimane nella regione servita ottimamente dalla stazione radio base 2, il circuito correlatore del cellulare tenterà di correlare ottimamente ($\langle X|P\rangle \rightarrow 1$) il codice $P_2(t)$ con il segnale $X_2(t)$ e di scorrelare altrettanto ottimamente ($\langle X|P\rangle \rightarrow 0$) $P_2(t)$ da tutti gli altri Y – 1 segnali codificati $X_i(t)$, ovvero da tutti gli altri Y – 1 codici di spreading $P_i(t)$. Stavolta soltanto la "copia" $X_2(t)$ della comunicazione costituisce il segnale codificato utile, quello gradito, mentre tutti gli altri Y – 1 segnali codificati $X_i(t)$, tutte le altre Y – 1 copie, provenienti dal medesimo cellulare trasmettitore, sono considerati come interferenze da reiettare quanto più possibile. Questa connessione in parallelo, nota come "macrodiversità", del terminale mobile, sia chiamante (N up-link in parallelo = codifica del segnale trasmesso da cellulare da parte di tutte le stazioni base dell'active set del cellulare stesso) che ricevente (N down-link in parallelo = ricezione della stessa informazione utile da più stazioni base contemporaneamente), a N stazioni radio base contemporaneamente (active set del trasmettitore = active set del ricevitore = N), e l'ortogonalità degli N codici di spreading $P_i(t)$, uno per ciascuna stazione radio base, ovvero uno per ciascun canale, consentono un approccio di tipo "soft handover" al problema del passaggio dei terminali mobili da una cella all'altra. La tecnica "soft handover" permette una migliore qualità della chiamata durante il movimento dei terminali mobili chiamanti e/o riceventi ed in particolare di minimizzare il rischio di interruzione della chiamata stessa nel passaggio (in $t = t_S$) del terminale mobile ricevitore (e/o trasmettitore) fra due celle contigue. Un approccio di tipo "hard handover" comporterebbe un maggior rischio di interruzioni e minore qualità della comunicazione, ma almeno non necessiterebbe di codici di spreading $P_i(t)$ fra loro perfettamente ortogonali nel tempo, in quanto il terminale mobile ricevitore effettuerebbe il down-link non da più stazioni radio base contemporaneamente, bensì da una sola, ovvero il proprio active set sarebbe pari ad 1 in qualunque istante. Ciò determinerebbe la possibilità di avere codici di canalizzazione non ortogonali (diciamo "quasi ortogonali"), ossia non perfettamente separabili: in altre parole il ricevitore si correla solo con il segnale X(t) dell'unica stazione radio base dalla quale arriva il segnale pilota, ignorando, per motivi di bassa potenza trasmessa, gli altri segnali codificati provenienti dalla stessa sorgente di segnale chiamante. Nei seguenti disegni si illustrano i concetti fin qui esposti:

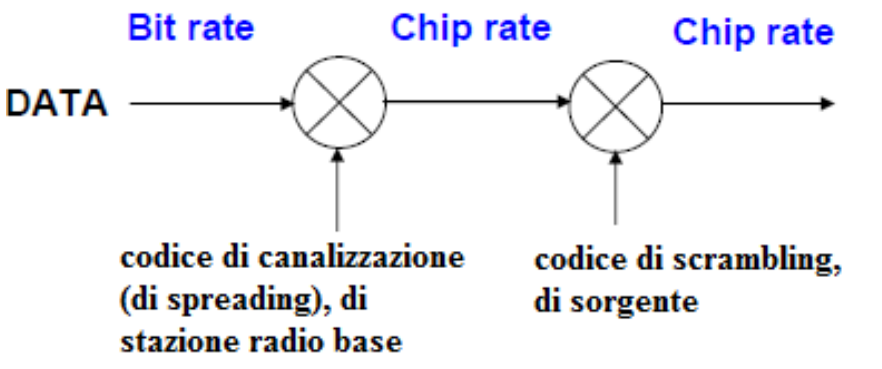


Consideriamo adesso un certo numero K di terminali mobili che trasmettono le rispettive stringhe informative di bit $S_i(t)$ (i = 1, 2, … , K), appoggiandosi ad una certa stazione radio base vicina. Questi K utenti del medesimo canale sono del tutto indipendenti gli uni dagli altri, i segnali che loro trasmettono sono del tutto asincroni (le K telefonate iniziano a K istanti diversi), pertanto se la stazione radio base si limitasse a moltiplicare ciascuno di questi K segnali digitali $S_i(t)$ per il codice pseudo-casuale di spreading $P_{SRB}(t)$, associato dalla stazione radio base a ciascuno dei K utenti (il codice di canalizzazione $P_{SRB}(t)$ è in realtà un codice caratteristico del canale, della stazione radio base, deciso dal settaggio dei bit di tap dello PNSG che serve ciascun canale, ed il settaggio del particolare G(x) primo dello PNSG è determinato dal sistema GPS), avremmo K sequenze pseudo-casuali asincrone $P_{SRBi}(t)$ (i = 1, 2, … , K), e in quanto asincrone (soggette pertanto a ritardi T non deterministici) assolutamente non ortogonali fra loro nel tempo, quindi non separabili ottimamente. Formalmente si ha che:

$$\left\langle P_{SRB_i}(t) \left| P_{SRB_j}(t) \right\rangle = \frac{1}{MT_{chip}} \int_{t = kT_{bit}}^{t = kT_{bit} + MT_{chip}} P_{SRB_i}(t)\ P_{SRB_j}(t)\ dt =\right.$$

$$= \langle P_{SRB}(t) \,|P_{SRB}(t+T)\rangle = \frac{1}{MT_{chip}} \int_{t = kT_{bit}}^{t = kT_{bit} + MT_{chip}} P_{SRB}(t)\ P_{SRB}(t+T)\ dt \neq 0, \text{ purtroppo.}$$

In altri termini le K sorgenti di segnale diverse, che condividono il canale comune costituito dalla stazione radio base di cui sopra, interferirebbero fra loro a livello di down-link in modo non trascurabile, ossia il circuito di despreading del cellulare ricevitore prima citato non riuscirebbe a selezionare, a "catturare", ottimamente (<X|P> → 1) la chiamata codificata ("chippata") desiderata, cioè quella dell'utente chiamante di cui si vuole ricevere la stringa di bit informativi, senza risentire di un'alta cross-correlazione, di un'alta interferenza, con le altre K – 1 telefonate, cioè con gli altri K – 1 segnali codificati $X_{SRBi}(t)$, provenienti dagli altri K – 1 utenti di quel canale. Pertanto per rendere distinguibili, separabili, questi K segnali provenienti da K sorgenti diverse, che utilizzano il canale comune rappresentato dalla stazione radio base, quest'ultima deve provvedere alla moltiplicazione di ciascun segnale $X_i(t)$ (i = 1, 2, … , K), già codificato con la sequenza pseudo-random $P_{SRB}(t)$ caratteristica del canale comune (che rende questi K segnali distinguibili dalle loro "copie" up-linkate su altri canali paralleli, ossia su altre stazioni radio base vicine), per un secondo codice pseudo-random $Scr_i(t)$, prodotto da un secondo PNSG, chiamato "codice pseudo-casuale di scrambling". I K codici di scrambling $Scr_i(t)$ (i = 1, 2, … , K) sono generati dalla stazione radio base comune in modo da essere ortogonali, pertanto dovrebbero essere molto lunghi, così da minimizzare i residui di cross-correlazione e rendere ben separabili le K utenze del canale comune. Quindi ciascun segnale digitale informativo trasmesso è codificato da due sequenza pseudo-casuali, quella di spreading, associata al canale utilizzato, cioè alla stazione radio base agganciata nell'up-link, e una di scrambling, associata alla sorgente, all'utente.

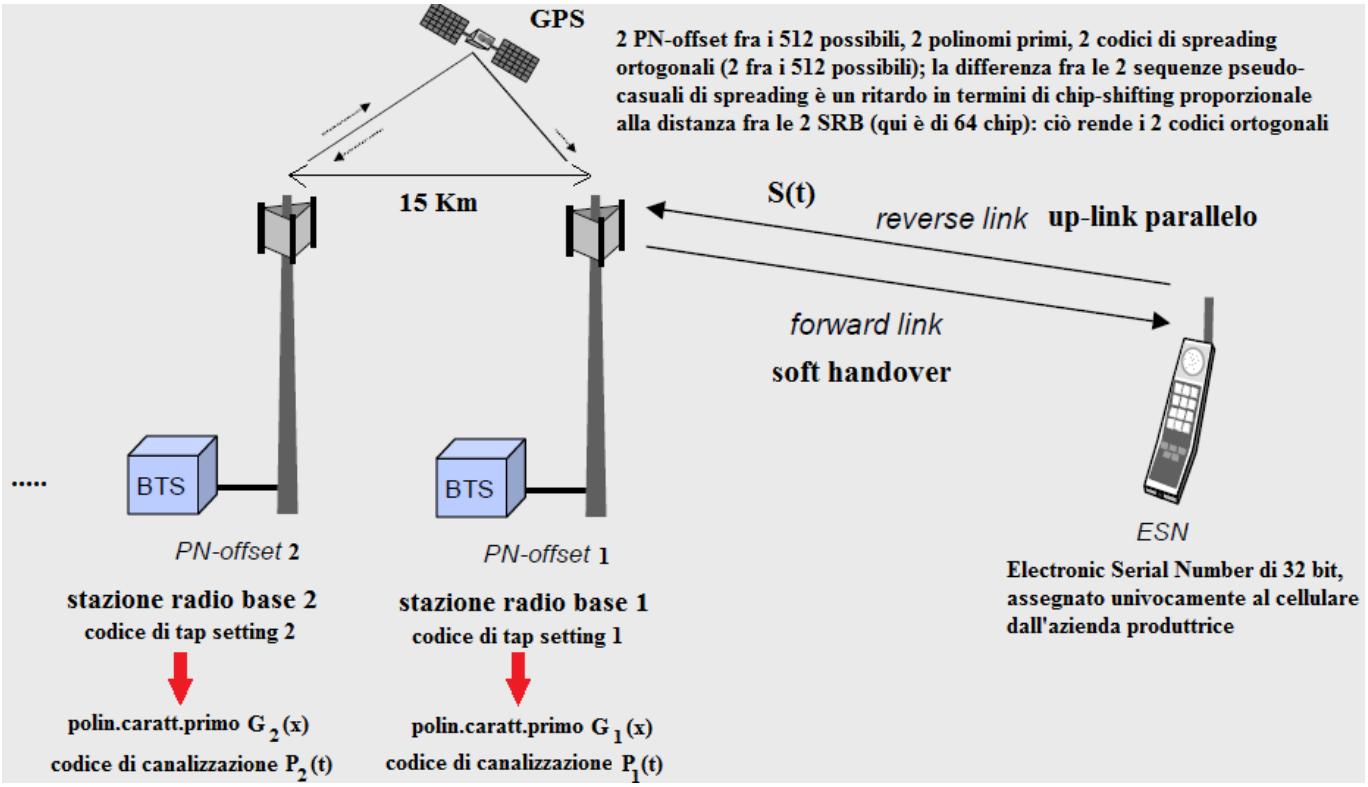


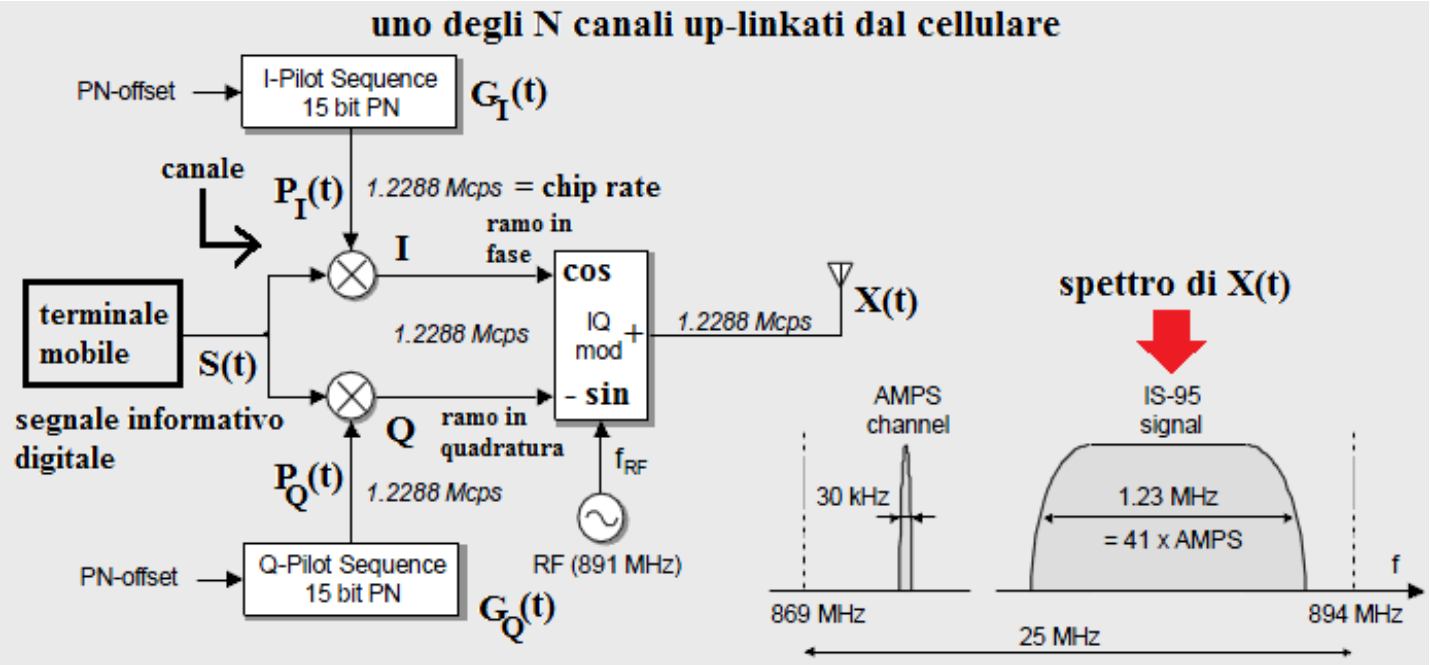
il codice di scrambling non altera lo Spreading Spectrum imposto ai dati dal codice di canalizzazione

Esistono ricevitori CDMA, chiamati ricevitori “Rake”, che per migliorare la qualità della chiamata non utilizzano soltanto, come già visto, i codici di spreading ortogonali, ma anche quelli di scrambling.
Sia nella tratta di up-link che in quella di down-link il sistema deve implementare algoritmi efficienti che supportino delle congrue politiche PC (“Power Control”, controllo di potenza): in fase di up-link il cellulare deve modulare la potenza irradiata in funzione delle variabili condizioni di propagazione (ostacoli, condizioni meteo…il canale “cambia” durante il movimento del terminale mobile), mentre in fase di down-link il cellulare deve capire qual’è, diciamo istante per istante, la stazione radio base più adatta, dal punto di vista della potenza trasmessa (vicinanza spaziale), con cui correlarsi. I controlli di potenza irradiata e ricevuta devono essere effettuati periodicamente: ad esempio lo standard CDMA IS – 95 prevede un controllo della potenza irradiata in fase di up-link, da parte del cellulare trasmettitore, effettuato con una frequenza di 800 Hz.

Standard CDMA IS – 95

Si tratta di un protocollo introdotto “ad interim” negli Stati Uniti nel 1995 (“IS – 95” sta per “Interim Standard 1995 ”, ne sono poi seguiti altri come IS – 97 e IS – 98) per le comunicazioni cellulari a codifica digitale (modulazione digitale). Questo protocollo utilizza di fatto la stessa banda di frequenze (869 – 894 MHz) usata dalle comunicazioni cellulari a modulazione analogica AMPS (“Advanced Mobile Phone System”).
Supponiamo di avere un utente (in realtà uno fra i tanti che possono avviare una trasmissione) che inizia a trasmettere la propria comunicazione, il proprio segnale digitale informativo S(t), e supponiamo che il cellulare agganci un certo numero N di stazioni radio base, ovvero che il terminale mobile trasmettitore effettui un up-link con N canali paralleli (macrodiversità). Nel disegno seguente è rappresentata, qualitativamente, una situazione in cui un utente trasmette una chiamata S(t) e questa aggancia N = 2 stazioni radio base.





Supponiamo che il segnale digitale informativo S(t) arrivi all'antenna della stazione radio base 1 in un certo istante t_1, a partire dal quale ciascun bit informativo di S(t) viene opportunamente codificato. Il canale rappresentato dalla stazione radio base 1 è infatti fornito di due PNSG a 15 stadi, ciascuno dei quali, settato opportunamente da una particolare combinazione dei 15 bit di tap (C_1… C_{15}, questa parola di settaggio è chiamata anche "PN – offset"), ovvero settato in modo da fornire un opportuno polinomio caratteristico primo G(x), e ricevuto in ingresso, come stato iniziale, il seme 000000000000001, fornisce in uscita una particolare keystream di spreading massimamente lunga, lunga cioè 491505 chip. Ciascuna delle due sequenze $P_I(t)$ e $P_Q(t)$ di canalizzazione va a battere, a mixare, ad allargare spettralmente, la rispettiva "copia" di S(t) (copia di fase e copia di quadratura). Il segnale chippato, codificato, $X_I(t)$ va a battere con la forma d'onda $\cos(2\pi f_0 t)$, mentre il segnale codificato $X_Q(t)$ va a battere con la forma d'onda $-\sin(2\pi f_0 t)$. Un sommatore provvede infine a sovrapporre nel tempo questi ultimi due segnali codificati e traslati in frequenza intorno alla radio-frequenza f_0 = 891 MHz. Il segnale finale codificato X(t) IS – 95, la cui banda è larga circa una quarantina di volte la banda tipica occupata da un segnale di telefonia wire-less modulato analogicamente e non digitalmente (secondo la tecnica GSM per esempio), è codificato da una particolare sequenza pseudo-casuale di chip.

Qualche istante dopo t_1, per esempio all'istante t_2, il segnale S(t) arriva anche alla stazione radio base 2. Il sistema GPS conosce la posizione delle due stazioni radio base, pertanto conosce il ritardo $t_2 - t_1$ con il quale S(t) inizia ad essere codificato, con le stesse identiche modalità "IQ – like", anche dalla seconda stazione radio base. Se i due PNSG della seconda stazione radio base producessero gli stessi identici codici $P_I(t)$ e $P_Q(t)$ di spreading caratteristici della prima stazione radio base, cioè se $G_I(t)$ e $G_Q(t)$ fossero gli stessi del primo canale, oppure se $P_I(t)$ e $P_Q(t)$ del secondo canale fossero semplicemente due codici appartenenti al set di codici ortogonali generabili da uno PNSG a 15 stadi, avremmo un evidente asincronismo fra il codice che identifica il segnale X(t), uscente dal modulatore IQ del primo canale, ed il codice che identifica il segnale Y(t), uscente dal modulatore IQ del secondo canale. Pertanto è necessario che i due PNSG del secondo canale vengano settati con bit di tap particolari, diversi da quelli che hanno settato i due PNSG del primo canale, ovvero che abbiano due G(x) diverse, quindi che producano due sequenze di

spreading, lunghe anch'esse 491505 chip, diverse da quelle del primo canale (PN – offset della stazione radio base 1 ≠ PN – offset della stazione radio base 2). In tal modo il segnale Y(t), ovvero la seconda copia "parallela" della telefonata (copia del segnale X(t)), è codificato in modo che la propria sequenza di chip sia la stessa sequenza di chip codificante X(t), shiftata però di un certo numero di chip, e tale numero di chip è proporzionale al ritardo $t_2 - t_1$ rilevato dal sistema GPS; questo ritardo dipende dalle posizioni relative di stazione radio base 1, stazione radio base 2 e terminale mobile. Nell'esempio riportato sopra le due stazioni radio base ed il cellulare si trovano posizionati lungo una retta e la distanza fra le due stazioni radio base è di 15 km, pertanto il ritardo di propagazione $t_2 - t_1$ ammonta a circa 52 μs, quindi la sequenza Y(t) che viaggia nel secondo canale è uguale a quella X(t) che viaggia nel canale parallelo, ossia nel primo canale, a meno di uno shift di 64 chip. Tale shift conserva l'ortogonalità fra X(t) e Y(t), nonostante il ritardo $t_2 - t_1$, che avrebbe dovuto, in linea di principio, rendere la cross-correlazione R_{xy} diversa da zero, comportando problemi di separabilità dei due segnali "copia" da stessa sorgente a livello di ricevitore.
A livello di down-link possiamo fare considerazioni analoghe a quelle già viste in precedenza. Un ricevitore in movimento, che adotta la tecnica del soft handover, dovrà capire quale stazione radio base, fra le K che in quegli istanti trasmettono in parallelo la stessa informazione utile, è la migliore dal punto di vista della potenza trasmessa, ovvero del segnale pilota (quale SRB serve meglio quella particolare cella in cui il cellulare ricevente si trova momentaneamente). Dopodiché il circuito correlatore deve correlarsi ottimamente con il segnale desiderato, ad esempio X(t), e scorrelarsi altrettanto bene con gli altri (Y(t)). La parte di back – end del sistema, che consente cioè il "forward link", è identica a quella di front – end che consente il "reverse – link". Il ricevitore IQ CDMA IS – 95 utilizzerà la stessa parola $C_1 \ldots C_{15}$ di settaggio PN – offset associata (mediante istruzione da parte del sistema GPS) a quel particolare canale che serve meglio, in quegli istanti, il cellulare ricevitore, per "istruire" i due PNSG a produrre le opportune sequenze pseudo-casuali di despreading, che consentano la demodulazione del segnale desiderato (della "copia" desiderata della comunicazione) e la reiezione degli altri segnali, sia dalla stessa sorgente che da sorgenti diverse.

Di seguito riportiamo il codice VHDL dello PNSG da noi progettato:

1 reg_FFD.vhd

```
1   ----------------------------------------------------------------------
2   -- (Behavioural)
3   -- File Name : dff.vhd
4   -- Purpose   : Positive Edge Triggered FlipFlopD-Asynchronous Reset
5   -- Library   : IEEE
6   -- Author(s) : Luca Santarelli, Enrico Molinari, Sandro Polliatti,
7   --                Stephane Tsomene Njousse
8   ----------------------------------------------------------------------
9
10  Library ieee;
11  use ieee.std_logic_1164.all;
12
13  Entity dff is
14      port( d    : in std_logic;
15            clk     : in std_logic;
16            reset : in std_logic;
17            q    : out std_logic );
18  end dff;
19
20
21  architecture Behavioural of dff is
22
23  begin
24
25   dff : process (clk)
26   begin             -- Reset attivo basso: finchè il reset
27                          -- è basso lo PNSG è disattivato,
28        if (reset = '0') then -- l'uscita q di ciascun FFD è mantenuta a zero,
29         q <= '0';       -- finchè reset = 0 in uscita dallo PNSG abbiamo
30        elsif (clk'event and clk = '1') then -- una sequenza PNcode di 15 zeri
31         q <= d;   -- Reset ad 1, l'uscita q di ciascun FFD viene
32        end if;   -- aggiornata con il valore dell'ingresso d
33    end process dff ;
34
35  end behavioural;
36
37
```

1 mux.vhd

```
1   ----------------------------------------------------------------------
2   -- (Behavioural)
3   -- File Name : mux.vhd
4   -- Purpose   : Multiplexer 2 to 1
5   -- Library   : IEEE
6   -- Author(s) : Luca Santarelli, Enrico Molinari, Sandro Polliatti,
7   --                Stephane Tsomene Njousse
8   ----------------------------------------------------------------------
9
10  Library ieee;
11  use ieee.std_logic_1164.all;
12
13  Entity mux is
14    port(  x0     : in std_logic;
15           x1   : in std_logic;
16           sel    : in std_logic;
17           z    : out std_logic );
18  end mux;
19
20  architecture Behavioural of mux is
21
22  begin
23      mux : process (x0,x1,sel)
24      begin                -- Sel basso, attivo canale x0:
25      if (sel = '0') then    -- l'uscita q dello stadio
26       z <= x0;           -- precedente è riportata sull'ingresso
27      else               -- d del FFD successivo
28       z <= x1;           -- Sel alto, attivo canale x1:
29      end if;             -- il corrispondente bit del seme è
30      end process mux;      -- riportato in uscita z al mux, ovvero
31                        -- in ingresso d al FFD servito dal mux
32  end behavioural;
33
34
```

1 PN_gen.vhd

```
1     ----------------------------------------------------------------------
2     -- (Structural)
3     -- File Name: Our_PnGen.vhd
4     -- Purpose  : Pseudo Noise Sequence Generator for CDMA IS-95
5     --          applications, phase branch
6     -- Library  : IEEE
7     -- Author(s): Luca Santarelli, Enrico Molinari, Sandro Polliatti,
8     --            Stephane Tsomene Njousse
9     ----------------------------------------------------------------------
10
11    Library ieee;
12    use ieee.std_logic_1164.all;
13
14    entity GenLess is
15        generic (N: INTEGER := 15);
16        port ( clk       :in std_logic;
17             init      :in std_logic;
18             IR         :in std_logic_vector (1 to N);
19             reset      :in std_logic;
20             PNcode    :out std_logic_vector(1 to N) );
21    end GenLess;
22        -- porte del dispositivo:
23        -- clk    : ingresso del clock per il PnGen pilotante i 15 FFD.
24        -- init   : Sul livello Alto inizializza l'ingresso d di ciascuno
25        --          dei 15 registri con il corrispondente bit della parola
26        --          di ingresso IR "Initial Register" (IR = seme = key);
27        --          PNSG ad anello aperto.
28        --          Sul livello Basso invece permetterà di chiudere
29        --          in loop cycle gli stadi del PnGen
30        -- IR     : Initial Register value = seme che lo PNSG espanderà
31        --           in un codice pseudo-casuale
32        -- Reset  : Attivo Basso, comanderà il reset di tutti
33        --           i 15 flip flop D
34        -- PNcode : codice pseudo-random in uscita dal dispositivo
35        --           su 15 bit (ad ogni clock cycle escono
36        --           15 bit del keystream), ciascuno dei quali
37        --           corrispondente all'uscita q
38        --           del rispettivo stadio dello PNSG
39
40    architecture structural of GenLess is
41
42    -- instanziamento di blocchi gerarchicamente inferiori
43    -- (già descritti comportamentalmente) e di segnali di appoggio
44
45        component dff
46           port (    d     :in std_logic;
47                 clk   :in std_logic;
48                 reset :in std_logic;
49                 q     :out std_logic );
50
51        end component;
52        -- Ogni stadio del PnGen
53        -- sarà composto da un Flip Flop D, la cui uscita q
54        -- entrerà in uno dei canali del mux.
55
56        -- Più precisamente durante la fase di settaggio con IR
57        -- l'uscita q del FFD i-esimo è collegata al canale x1 del mux
58        -- i-esimo che serve quel particolare FFD, ovvero al bit IR(i);
59        -- durante la fase di generazione di codici pseudo-casuali
60        -- q del FFD i-esimo è collegata al canale x0 del mux
```

```
61        -- i-esimo, ovvero all'uscita q del FFD i-1-esimo
62
63        component mux
64           port (    x0    :in std_logic;
65                     x1    :in std_logic;
66                     sel   :in std_logic;
67                     z     :out std_logic  );
68        end component;
69
70        -- la porta sel di ciascun mux è collegata con la porta
71        -- init dello PNSG (blocco gerarchicamente superiore),
72        -- e tramite questa si comandano contemporaneamente
73        -- gli stati di tutti i 15 mux interni
74
75
76        signal qmux : std_logic_vector (1 to N);
77        -- segnale di appoggio (bus di 15 bit): ciascuno dei
78        -- suoi 15 bit riporta l'uscita q di ciascun
79        -- flip flop D in ingresso al mux successivo
80
81        signal inXor : std_logic_vector (1 to 6);
82        -- segnale di appoggio (bus di 6 bit) che collega
83        -- le uscite q degli stadi 5-7-8-9-13-15
84        -- (coefficienti non nulli del polinomio caratteristico G(x))
85        -- agli ingressi delle porte xor, secondo
86        -- la topologia circuitale descritta nella relazione
87
88        signal muxd : std_logic_vector (1 to N);
89        -- segnale di appoggio (bus di 15 bit): ciascuno dei suoi
90        -- 15 bit riporta l'uscita z di ciascun mux
91        -- sull'ingresso d del flip flop D successivo
92
93        signal outXor : std_logic_vector (1 to 5);
94        -- segnale di appoggio (bus di 5 bit): ciascuno dei suoi
95        -- 5 bit collega l'uscita di ciscuna delle 5 porte xor
96        -- verso la porta successiva secondo la topologia scelta
97
98
99        BEGIN
100       -- Generazione della catena diretta (A) del circuito PNgen
101       -- per mezzo di un ciclo for - generate.
102
103       -- Si distinguono sostanzialmente 3 casistiche:
104
105       GEN_structure: for i in 1 to N generate
106
107       --Caso i=1: Se il dispositivo è nella fase di
108       --         installazione dell'IR (init=1), cioè
109       --          di inserimento del seme (caricamento
110       --          dello stato iniziale), si avrà il valore
111       --          di IR(1), in ingresso al canale x1 del mux(1),
112       --          riportato sull'uscita z del mux(1),
113       --          altrimenti se init=0 si avrà
114       --          l'uscita della xor5, in ingresso al
115       --          canale x0 del mux(1), riportata
116       --          sull'uscita z del mux(1).
117       --          Lo Stadio 1 si distingue da tutti gli altri poichè
118       --          sarà responsabile della chiusura dell'anello
119       --          di reazione (catena "beta") del circuito PNgen.
120
121          FIRST: if i = 1 generate
122            MX1: mux port map -- instanziamento del mux1
123             ( x0     => outXor(5),
```

```
124             x1     => IR(i),
125             sel    => init,
126             z      => muxd(i) );
127
128           FF1 : dff port map -- instanziamento dello stage1(FFD1)
129            ( d         => muxd(i),
130              clk       => clk,
131              reset     => reset,
132              q         => qmux(i) );
133
134             PNcode(i) <= qmux(i);
135
136        end generate FIRST;
137
138     -- Caso 1<i<N: Se il dispositivo è nella fase di
139     --             installazione dell'IR (init=1),
140     --              allora si avrà il valore
141     --              di IR(i) in ingresso al canale x1 del mux(i).
142     --              In corrispondenza di init=0,
143     --              ciascuno stadio i-esimo
144     --              raccoglierà sul relativo mux(i),
145     --              sul canale x0, l'output q dello
146     --              stadio i-1-esimo e aggiornerà,
147     --              ad ogni fronte di salita del clock,
148     --              l'uscita q del proprio flip flop D
149     --              con il valore dell'ingresso d
150
151     internal: if i > 1 and i < N generate
152     -- instanziamento di mux2, mux3...mux14
153           MXi: mux port map
154            ( x0     => qmux(i-1),
155              x1     => IR(i),
156              sel    => init,
157              z      => muxd(i) );
158  -- instanziamento di stage1(FFD1),stage2(FFD2)...stage14(FFD14)
159           FFi : dff port map
160            ( d         => muxd(i),
161              clk       => clk ,
162              reset     => reset,
163              q         =>qmux(i) );
164
165           PNcode(i) <= qmux(i);
166
167        end generate internal;
168
169     --Caso i=N : Se init=1 accadrà, analogamente a tutti
170     --           gli altri stadi, di avere in ingresso
171     --           al canale x1 del mux(N) il bit
172     --           IR(N) del seme (MSB del seme).
173     --           Se init=0, lo stadio N raccoglierà
174     --           l'uscita q del flip flop D
175     --           dello stadio precedente
176     --           (stage14) e la riporterà in ingresso alla
177     --           porta xor1, la prima della
178     --           serie delle 5 porte
179     --           xor responsabili dell'implementazione
180     --           del polinomio caratteristico G(x) del
181     --           PNgen in questione.
182     --           La peculiarità di questo stadio consiste
183     --           nell'avviare l'anello di retroazione,
184     --           ovvero di alimentare, con l'uscita q,
185     --           la prima xor dell'anello di feedback,
186     --           anzichè alimentare, come i precedenti
```

```
187       --           14 stadi, il canale x0 di un mux
188
189       LAST: if i = N generate
190          -- instanziamento dell'ultimo mux
191            MXn: mux port map
192             ( x0     => qmux(i-1),
193               x1     => IR(i),
194               sel    => init,
195               z      => muxd(i) );
196          -- instanziamento dell'ultimo FFD
197            FFn : dff port map
198             ( d         => muxd(i),
199               clk       => clk ,
200               reset     => reset,
201               q         => inXor(1) );
202
203            PNcode(i) <= inXor(1);
204
205          end generate LAST;
206
207       end generate GEN_structure;
208
209
210       -- La seguente parte di codice
211       -- descrive la topologia circuitale
212       -- dell'anello di feedback,
213       -- cioè della catena "beta" di retroazione
214       -- (il "piano" delle porte xor),
215       -- responsabile dell'implementazione
216       -- del polinomio caratteristico G(x)
217       -- del nostro PNSG.
218       -- Si è scelta una soluzione ad hoc,
219       -- cioè di tipo ASIC, per il particolare
220       -- polinomio G(x) = 1+x^5+x^7+x^8+x^9+x^13+x^15,
221       -- ponendo le porte xor solo in corrispondenza
222       -- delle uscite q degli stadi posizionati
223       -- secondo gli esponenti del polinomio:
224       --   OutPortXor1 = q15 XOR q13;
225       --   OutPortXor2 = OutPortXor1 XOR q9;
226       --   OutPortXor3 = OutPortXor2 XOR q8;
227       --   OutPortXor4 = OutPortXor3 XOR q7;
228       --   OutPortXor5 = OutPortXor4 XOR q5;
229
230
231       OutPortXor1:
232       inXor(2) <= qmux(13);
233       outXor(1) <= inXor(1) xor inXor(2);
234       -- -- inXor(1) <= q dell'ultimo FFD;
235       -- OutPortXor1 = q15 XOR q13;
236
237       OutPortXor2:
238       inXor(3) <= qmux(9);
239       outXor(2) <= outXor(1) xor inXor(3);
240       -- OutPortXor2 = OutPortXor1 XOR q9;
241
242       OutPortXor3:
243       inXor(4) <= qmux(8);
244       outXor(3) <= outXor(2) xor inXor(4);
245       -- OutPortXor3 = OutPortXor2 XOR q8;
246
247       OutPortXor4:
248       inXor(5) <= qmux(7);
249       outXor(4) <= outXor(3) xor inXor(5);
```

```
250       -- OutPortXor4 = OutPortXor3 XOR q7;
251
252       OutPortXor5:
253       inXor(6) <= qmux(5);
254       outXor(5) <= outXor(4) xor inXor(6);
255       -- OutPortXor5 = OutPortXor4 XOR q5;
256
257
258   end structural ;
259
```

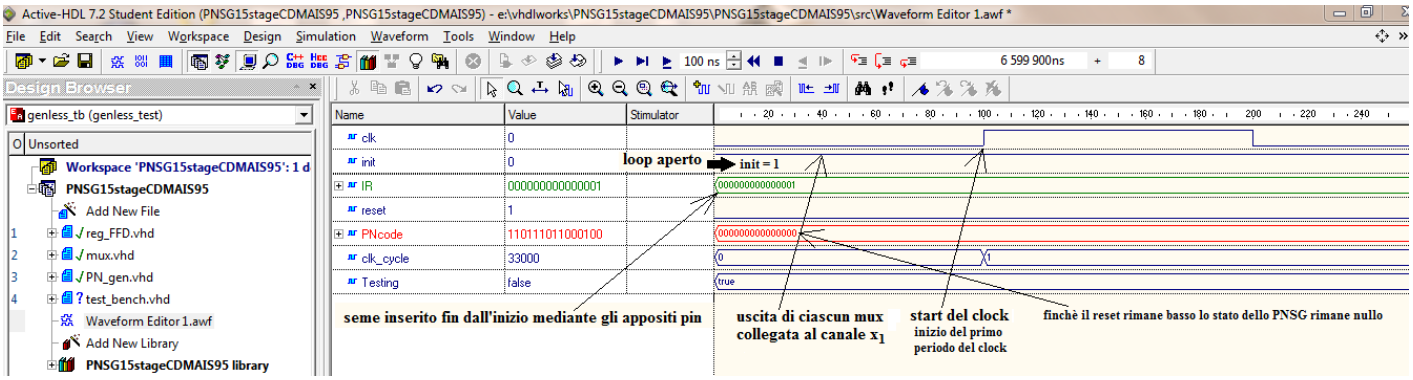
1 test_bench.vhd

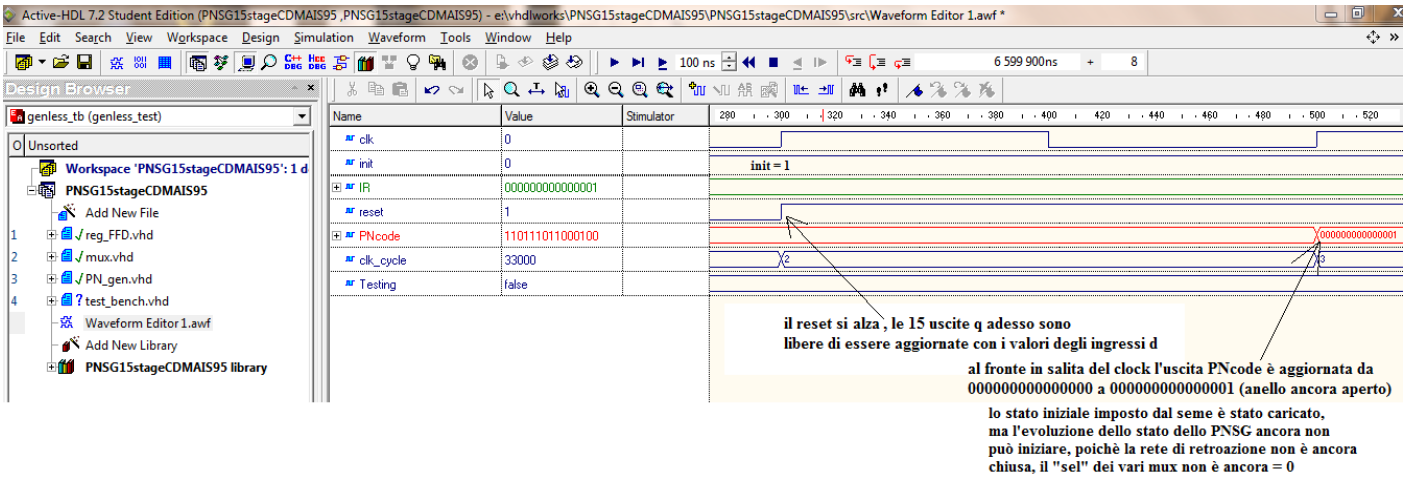
```vhdl
    ---------------------------------------------------------------------------
    -- (Structural)
    -- File Name : Our_PnGenTB.vhd
    -- Purpose   : to generate Stimuli for Pseudo Noise Sequence Generator
    --             in CDMA IS - 95 applications, in order to proof periodicity
    --             of pseudo-random sequence
    -- Library   : IEEE
    -- Author(s) : Luca Santarelli, Enrico Molinari, Sandro Polliatti,
    --                Stephane Tsomene Njousse
    ---------------------------------------------------------------------------

    Library ieee;
    use ieee.std_logic_1164.all;

    entity Genless_tb is -- empty entity
    end Genless_tb;

    architecture genless_test of Genless_tb is
     -- architettura strutturale
    ---------DUT: device (design) under test----------
    component Genless

         generic (N: INTEGER := 15);
         port    ( clk     :in std_logic;
                   init    :in std_logic;
        -- init attivo alto (inizializza i 15 registri con IR = seme)
                  IR      :in std_logic_vector (1 to N);
                  reset   :in std_logic;
                   PNcode  :out std_logic_vector(1 to N)  );

    end component GenLess;

    -------------constants-------------------

    CONSTANT MckPer  : TIME    := 200 ns;
    --master clock period
    CONSTANT TestLen : integer := 66000;
    --number of count for test (MckPEr/2)

    -- Si osserva il comportamento del dispositivo
    -- su di un numero di periodi
    -- di clock molto elevato al fine di verificare
    -- la periodicità della sequenza
    -- pseudo-casuale PNcode. In effetti ci aspettiamo,
    -- per ragioni legate alla natura
    -- dell'algoritmo che lo PNSG implementa,
    -- una periodicità di 32767 fronti di clock in salita.
    -- Poichè il numero di periodi è misurato in termini
    -- di semiperiodi MckPer/2, allora ci aspettiamo
    -- che 32768 fronti in salita di clock
    -- si verifichino nell'arco di circa 65536
    -- semiperiodi. Si arrotonda a 66000 semiperiodi
    -- per verificare che la periodicità non
    -- solo sussista (ritorno allo stato iniziale "IR"),
    -- ma che si mantenga (noi lo verifichiamo
    -- per un ragionevole numero di semiperiodi).

    CONSTANT N : INTEGER := 15  ; --SHIFT REGISTER LENGHT

```

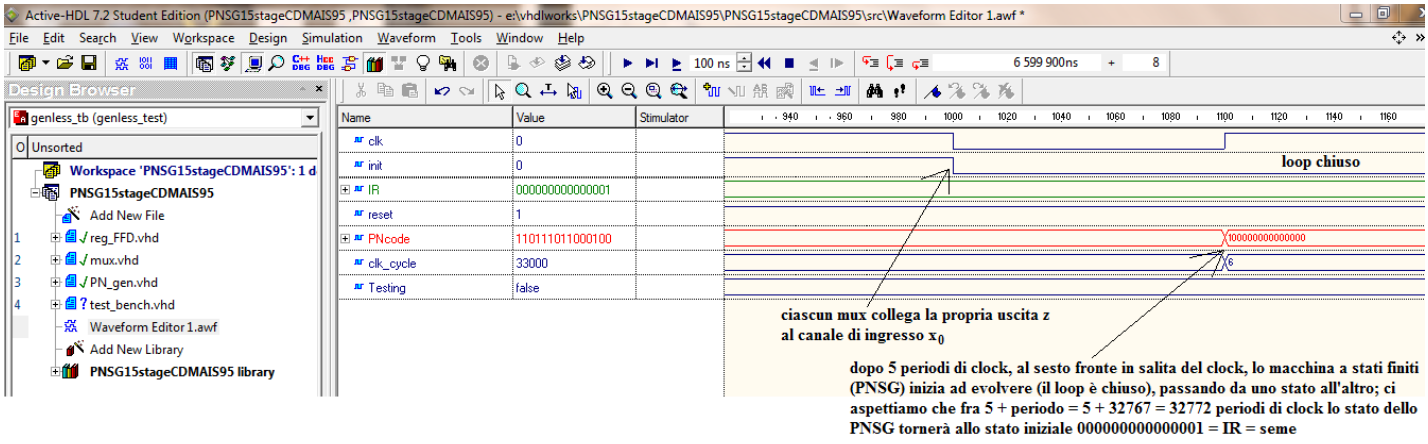
```
61    --------------INPUT SIGNALS---------------
62
63    SIGNAL clk : std_logic := '0';
64    -- clock inizialmente a 0
65
66    SIGNAL init : std_logic := '1';
67    -- init a 1 (sel = 1 su ciascuno dei 15 mux),
68    -- cioè PNgen in fase di installazione IR
69    -- finchè init = 1 il loop dello PNSG è aperto
70
71    signal IR : std_logic_vector (1 to N):= "000000000000001";
72    -- si sceglie questo seme. Questa scelta ci
73    -- permette di generare una sequenza massimamente lunga,
74    -- cioè di lunghezza pari al periodo = 2^15 - 1 = 32768,
75    -- essendo G(x) primo
76
77    signal reset : std_logic := '0';
78    --i flip flop sono inizialmente inattivi,
79    -- i 15 q sono imposti a zero
80
81
82    -------------OUTPUT & special SIGNALS-----------
83
84    signal PNcode : std_logic_vector (1 to N);
85
86    signal clk_cycle : integer; -- contatore dei cicli di clock
87    signal Testing : Boolean := True;
88
89    begin
90
91        I : Genless generic map (N)
92           -- instanziamento (local binding) del DUT
93                    port map (clk,init,IR,reset,PNcode);
94           -- notazione posizionale
95
96    ------------------------------------------------
97    ------------------generate clk----------------
98        clk <= NOT clk AFTER MckPer/2 when Testing else '0';
99
100
101          Test_Proc : PROCESS (clk)
102
103          variable count : INTEGER := 0;
104
105          BEGIN
106            clk_cycle <= (count+1)/2;
107
108            case count is
109             when 3 => reset <= '1';
110
111       -- Al terzo fronte del clock, cioè al secondo fronte in salita
112       -- del clock (inizio del secondo ciclo di clock),
113       -- ovvero dopo 300ns, il reset dei 15 flip flop
114       -- passa ad 1, attivandone le uscite q.
115       -- Indipendentemente dal valore del reset
116       -- sui flip flop, di default si è posto
117       -- init = 1 e questo significa che si impone (si applica)
118       -- fin da subito il seme sugli ingressi d
119       -- dei flip flop.
120
121                    when 10 => init <= '0';
122              -- chiusura del loop di feedback, inizio dell'evoluzione
123              -- dello stato dello PNSG
```

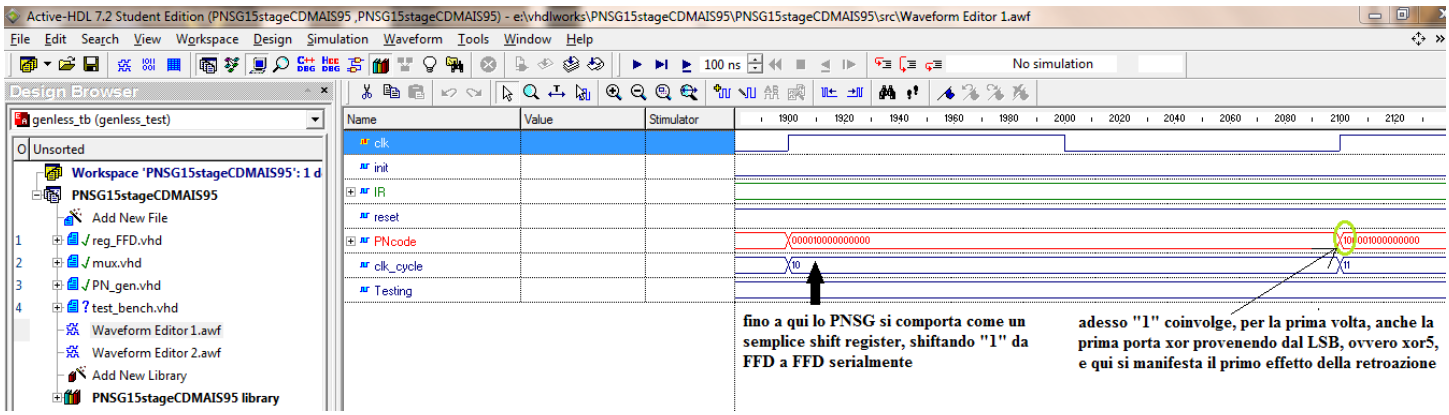
```
124
125
126             when (TestLen-1) => Testing <= False;
127             when others => NULL;
128            end case;
129
130   -- Con questa temporizzazione delle
131   -- variazioni degli stimoli (reset, init...)
132   -- si verifica una ripetizione dell'IR
133   -- al 32772-esimo colpo di clock.
134   -- I 5 colpi di clock in eccesso,
135   -- rispetto alla periodicità prevista (32767),
136   -- sono dovuti ai primi 5 fronti in salita del clock,
137   -- in corrispondenza dei quali l'init persisteva
138   -- sul valore alto (anello di retroazione aperto).
139   -- l'anello di feedback si chiude al sesto ciclo di clock
140   -- e dopo 32767 cicli di clock (si arriva al 32772-esimo ciclo)
141   -- la sequenza PNcode di uscita ritorna al valore IR = seme
142
143            count := count + 1;
144
145          End Process Test_Proc;
146
147   End genless_test;
```

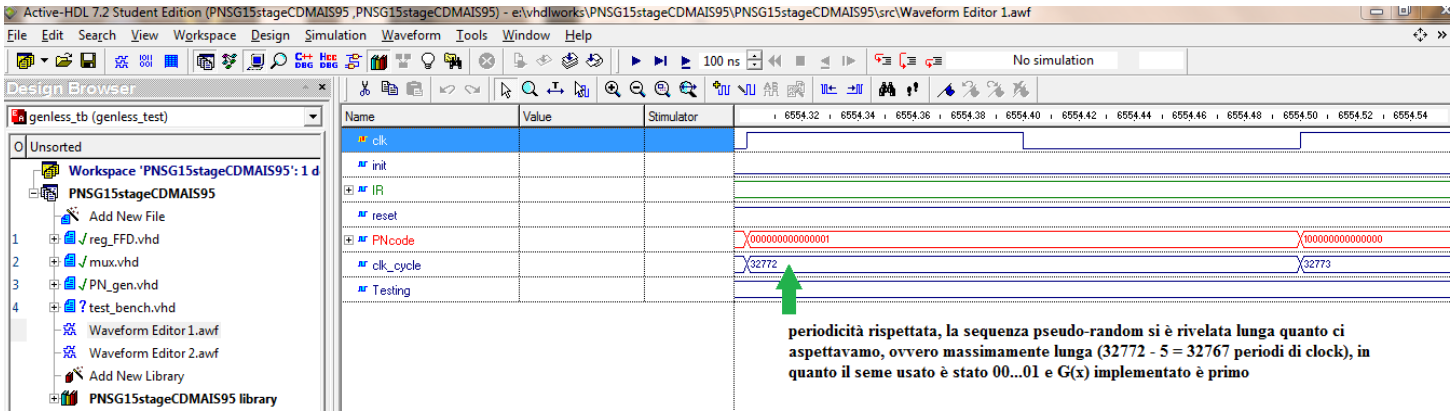
Alla fine della simulazione abbiamo osservato l'andamento temporale dei segnali di ingresso e uscita utilizzati nel test bench, di cui riportiamo, qui di seguito, alcune porzioni significative:








Active-HDL 7.2 Student Edition (PNSG15stageCDMAIS95 ,PNSG15stageCDMAIS95) - e:\vhdlworks\PNSG15stageCDMAIS95\PNSG15stageCDMAIS95\src\Waveform Editor 1.awf
File Edit Search View Workspace Design Simulation Waveform Tools Window Help
100 ns
No simulation
Design Browser
genless_tb (genless_test)
Unsorted
Workspace 'PNSG15stageCDMAIS95': 1 d
PNSG15stageCDMAIS95
Add New File
reg_FFD.vhd
mux.vhd
PN_gen.vhd
test_bench.vhd
Waveform Editor 1.awf
Waveform Editor 2.awf
Add New Library
PNSG15stageCDMAIS95 library
Name
Value
Stimulator
clk
init
IR
reset
PNcode
clk_cycle
Testing
fino a qui lo PNSG si comporta come un semplice shift register, shiftando "1" da FFD a FFD serialmente
adesso "1" coinvolge, per la prima volta, anche la prima porta xor provenendo dal LSB, ovvero xor5, e qui si manifesta il primo effetto della retroazione


Active-HDL 7.2 Student Edition (PNSG15stageCDMAIS95 ,PNSG15stageCDMAIS95) - e:\vhdlworks\PNSG15stageCDMAIS95\PNSG15stageCDMAIS95\src\Waveform Editor 1.awf
File Edit Search View Workspace Design Simulation Waveform Tools Window Help
100 ns
No simulation
Design Browser
genless_tb (genless_test)
Unsorted
Workspace 'PNSG15stageCDMAIS95': 1 d
PNSG15stageCDMAIS95
Add New File
reg_FFD.vhd
mux.vhd
PN_gen.vhd
test_bench.vhd
Waveform Editor 1.awf
Waveform Editor 2.awf
Add New Library
PNSG15stageCDMAIS95 library
Name
Value
Stimulator
clk
init
IR
reset
PNcode
clk_cycle
Testing
periodicità rispettata, la sequenza pseudo-random si è rivelata lunga quanto ci aspettavamo, ovvero massimamente lunga (32772 - 5 = 32767 periodi di clock), in quanto il seme usato è stato 00...01 e G(x) implementato è primo